universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Domain-level design and simulation of pseudoknot-free cotranscriptional folding pathways

verfasst von | submitted by

Laurenz Miksch BSc

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 875
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Bioinformatik
Betreut von | Supervisor:	Univ.-Prof. Dipl.-Phys. Dr. Ivo Hofacker

Acknowledgements

I would like to thank Prof. Ivo Hofacker for giving me the opportunity to work at the TBI and and for supporting my work on this project.

Thanks to everyone at the TBI for providing such a pleasant work environment, especially Prof. Hofacker and Prof. Christoph Flamm for fostering this atmosphere.

A special thanks to Stefan Badelt, PhD for guiding me through this journey. It has been a pleasure working together and discussing the scientific details of this thesis. Without your support, this would not have been possible.

I would also like to thank my parents, Susi and Gerhard, for making this all possible and supporting my academic journey. Thanks to Lena for always being there for me. Finally, I would like to thank the rest of my family and friends for their continued support.

Kurzfassung

RNA-Moleküle falten sich während ihrer Synthese zu komplexen Strukturen, ein Prozess, der als cotranskriptionelle Faltung bezeichnet wird. Der Verlauf dieses Prozesses kann die Struktur und Funktion der RNA entscheidend beeinflussen. Daher ist es wichtig, RNAs nicht nur im Hinblick auf ihre endgültige Struktur zu designen, sondern den Prozess der cotranskriptionellen Faltung bereits im Design gezielt zu berücksichtigen und während der Transkription aktiv zu steuern. Diese Masterarbeit stellt eine neue Methodik zum Design von RNA-Sequenzen auf der Basis sogenannter **abstrakter cotranskriptioneller Faltungspfade (aCFPs)** vor. Ein aCFP definiert eine vordefinierte Abfolge von Strukturen, die die RNA zu bestimmten Zeitpunkten während der Transkription einnehmen soll und führt das RNA-Molekül schrittweise zu seiner endgültigen Struktur. Dabei konzentrieren sich aCFPs ausschließlich auf strukturelle Veränderungen von Helices und ungepaarten Bereichen, ohne explizite Helixlängen oder detaillierte Sequenzanforderungen berücksichtigen zu müssen. Um aCFPs effizient zu implementieren, wird in dieser Arbeit eine Domänenebene als Zwischenschritt verwendet, um die abstrakte Design-Ebene mit der konkreten Nukleotidebene zu verbinden. Inspiriert durch domain-level strand displacement systems integriert das hier vorgestellte Designschema spezifische kurze Domänen (Toeholds), um gezielt strukturelle Umfaltungen während der Transkription auszulösen. Die Domänenebene wird dabei nicht nur als Design-Zwischenschritt genutzt, sondern dient gleichzeitig als Validierungsschritt. So wird die cotranskriptionelle Faltung zunächst auf Domänenebene simuliert und überprüft, bevor auf Nukleotidebene weitergearbeitet wird. Im Rahmen dieser Arbeit wurde das Softwarepaket **CocoPaths** entwickelt, welches drei Programme für das Design und die Validierung von cotranskriptionellen Faltungspfade umfasst: (I) **CocoPath** validiert aCFPs und übersetzt sie in domänenbasierte Sequenzen; (II) **CocoSim** simuliert die cotranskriptionelle Faltung auf Domänenebene und ermöglicht die Überprüfung und Optimierung der entworfenen Sequenzen; (III) **CocoDesign** übersetzt validierte domänenbasierte Designs in konkrete nukleotidbasierte Sequenzen, die den gewünschten Faltungspfade auf Nukleotidebene realisieren. Durch die Einführung dieses domänenbasierten Ansatzes und der zugehörigen Software CocoPaths erweitert diese Masterarbeit bestehende RNA-Designmethoden und bietet eine solide Grundlage für das Design und die Validierung von RNA-Sequenzen mit gezielt kontrollierbaren cotranskriptionellen Faltungspfade.

Abstract

RNA molecules fold into complex structures during synthesis in a process known as cotranscriptional folding. The path taken during this folding can critically influence RNA structure and function, making it essential not only to design specific RNA structures but also to control how they form during transcription. This thesis introduces a novel methodology for designing RNA sequences based on **abstract cotranscriptional folding paths (aCFPs)**. An aCFP represents a predefined series of structures that the RNA should adopt at specific points during transcription, guiding the RNA molecule step-by-step toward its final structure. The aCFP corresponds exclusively to large-scale structural transitions of helices and unpaired regions, bypassing the need to consider helix lengths or specific sequence constraints. In order to effectively implement aCFPs, this thesis uses a domain-level abstraction as an intermediate specification layer, bridging the conceptual gap between aCFPs and precise nucleotide-level sequences. Inspired by domain-level strand displacement systems, the design approach integrates short domains acting as toeholds to initiate structural rearrangements during transcription. By employing domains, the design process simplifies the complexity of sequence generation and significantly enhances the ability to validate designs at the domain-level before committing resources to detailed nucleotide-level optimization. In parallel with the development of the theoretical framework, a software package called **CocoPaths** was created, which includes three tools for designing and verifying folding paths: **(I) CocoPath**, for designing domain level sequences based on an aCFP; **(II) CocoSim**, a domain level cotranscriptional folding simulator for sequence verification and optimization; and **(III) CocoDesign**, which translates validated domain level designs into nucleotide sequences that exhibit the desired cotranscriptional folding path at the nucleotide level. By introducing this domain level framework and the CocoPaths software suite, this thesis advances RNA design methodologies, providing a robust foundation for the design and validation of cotranscriptionally interesting folding pathways.

Contents

1 Background

1.1 RNA fundamentals

Ribonucleic acid (RNA) is one of the most important biomolecules in living organisms. Its most commonly known function is serving as an intermediate between DNA and proteins, facilitating protein synthesis. However, only about 2% of the RNA produced in a cell is used for protein synthesis (coding RNA), the remaining 98% perform various functions at the RNA level as non-coding RNAs [1]. Understanding the functions of non-coding RNAs is therefore of great interest. Since the functions of biomolecules are often linked to their structures, enhancing RNA structural predictions is essential for deepening our understanding of RNAs.

Ribonucleic acid (RNA) is a biopolymer consisting of nucleotides. Each nucleotide consists of a base, a ribose sugar, and a phosphate group. The nucleotides are linked by phosphodiester bonds between the ribose sugar of one nucleotide and the phosphate group of the next, forming a single-stranded molecule with a negatively charged backbone of ribose sugars and phosphate groups. RNA nucleotides primarily contain four different bases: adenine (A), guanine (G), cytosine (C), and uracil (U). These four bases are called canonical bases and form canonical base pairs. Hydrogen bonds and stacking between the bases are fundamental to the formation of three-dimensional structures. Adenine and Uracil form two hydrogen bonds, while guanine and cytosine form three hydrogen bonds. The formation of these hydrogen bonds between complementary bases leads to the development of RNA's three-dimensional structures.
Similar to proteins, RNA structures can be described at four different levels. The primary structure provides information about the order of the nucleotides in the sequence. The secondary structure includes the base pairs that can be mapped onto a two-dimensional plane. The tertiary structure refers to the three-dimensional arrangement of the RNA molecule, incorporating interactions between secondary structure motifs such as helices. The quaternary structure involves interactions with proteins or other RNAs.

In the secondary structure, the base pairs are usually Watson-Crick pairs (isosteric base pairs), including A-U and G-C pairs, as well as wobble base pairs like G-U. Additionally, non-canonical base pairs can form, although these are energetically weaker compared to canonical base pairs. [2] Consecutive base pairs result in base stacking, where the bases are stacked on top of each other, a phenomenon known as π-stacking, which further stabilizes the RNA structure.[3]

This thesis focuses exclusively on sequences and secondary structures containing canonical bases. Therefore, the following definitions are provided:

Definition 1. *A **nucleotide sequence** $\eta = [\eta_1, \eta_2, \ldots, \eta_n]$ is a list of length n, where $\eta_i \in \{A, C, G, U\}$. By convention, RNA nucleotide sequences are given in 5' to 3' orientation.*

Definition 2. *A **nucleotide secondary structure** for a sequence η is a set of base-pair indices $S = \{(i, j)\}$ that satisfies the following conditions:*

- ***Permissible Base-Pairing:** For each $(i, j) \in S$, the base pair (η_i, η_j) must be one of the following pairs: $(A, U), (U, A), (C, G), (G, C), (G, U), (U, G)$.*
- ***Uniqueness of Pairing:** Each nucleotide can form at most one base pair. This means if $(i, j) \in S$ and $(i, k) \in S$, then $j = k$.*
- ***Nesting Condition:** Pairs must be nested. For any two pairs $(i, j), (p, q) \in S$, if $i < p < j$, then it must hold that $i < q < j$.*
- ***Minimum Hairpin Loop Length:** A hairpin loop must contain at least three unpaired nucleotides. Thus, if $(i, j) \in S$, then $|j - i| \geq 3$.*

There exists no universal annotation for representing RNA secondary structures, as there are many advantages and disadvantages to different annotations. This thesis uses primarily the dot-bracket annotation to represent nucleotide secondary structures. The **dot-bracket annotation**, used in the ViennaRNA package[4], is one of the most common ways of representing secondary structures in the form of characters in a string. Each character in the string represents one nucleotide. An unpaired nucleotide is represented by a dot, while paired nucleotides are represented by brackets. This representation is preferred when structures are used as input for algorithms, since paired nucleotides in a structure can be easily converted to an adjacency matrix.

1.1.1 RNA structural features

On the secondary structure level, RNAs can exhibit various structural motifs [5]. The following motifs are relevant for this thesis and are illustrated in Figure 1.1:
Helix: A helix is a region of consecutive base pairs. Due to the phenomenon of base stacking that occurs between pairs, helices form a stable structure. They are a primary factor driving the form and function of RNAs.
Stem-loop: A stem-loop or hairpin-loop consists of a helix capped by an unpaired loop region at one end. According to Definition 2, the loop must contain at least three unpaired nucleotides, meaning the loop must have a length greater or equal than 3.
Pseudoknots: According to Definition 2, base pairs must be nested. Crossing base pairs, where following base pairs (i, j) and (p, q) are paired such that $i < p < j < q$, are defined as pseudoknots. Generally, thermodynamic and kinetic modeling of pseudoknots is lagging behind, as crucial energy parameters are still missing. Therefore, the structural space considered in this thesis is limited to non-pseudoknotted structures.
Four-way junction: A four-way junction involves four double-stranded RNA helices intersecting at a single point, creating an X-shaped configuration. [6]

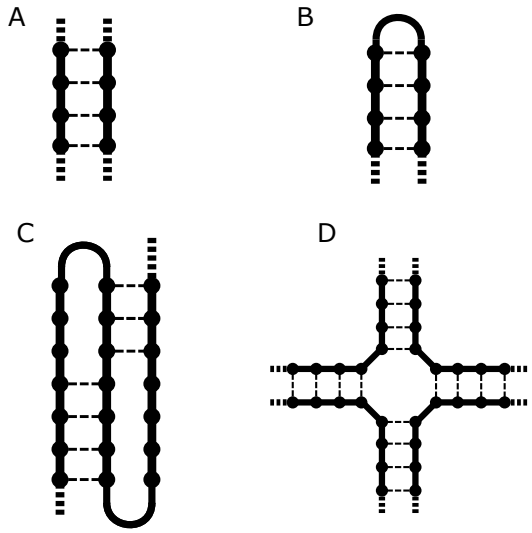


Figure 1.1: Visualization of RNA secondary structural features. Each node represents a nucleotide, and dotted lines indicate base-pair interactions between nucleotides. The following structures are depicted: (A) Helix, (B) Stem-loop, (C) Pseudoknot, and (D) Four-way junction.

1.2 Thermodynamics and kinetics of RNA folding

Understanding and predicting the structural features of RNA requires a framework that incorporates both thermodynamics and kinetics of RNA folding. Thermodynamic models involve the use of energy calculations to predict RNA structures, assuming that the molecule will adopt the most energetically favorable conformation. This approach considers only a snapshot of the RNA molecule, focusing on the most probable or thermodynamically stable structure without accounting for the dynamics of folding over time.
In contrast, kinetic approaches to RNA secondary structure prediction recognize the inherent dynamism of RNA molecules, where base pairs can form and break spontaneously, or where new nucleotides are added sequentially during transcription. RNA folding is influenced by the sequential synthesis of the RNA strand and the transient intermediate structures that emerge during this process. Kinetic folding predictions simulate fundamental structural changes and reactions that occur as the RNA sequence grows and folds. These elementary steps include actions such as the opening and closing of base pairs, allowing the model to capture the time-dependent pathways and transient structures that emerge during transcription. Beyond transcription, kinetic factors also play a crucial role in structural rearrangements, such as those seen in riboswitches, where changes in structure are driven by specific molecular interactions or environmental cues, allowing RNA to dynamically alter its conformation in response to functional needs.

1.2.1 Thermodynamic models for RNA structures

Thermodynamic models provide a foundational framework for predicting RNA secondary structures by calculating the energies associated with different configurations. A commonly used model for RNA folding predictions is the nearest-neighbor model. This model decomposes down the RNA secondary structure into its constituent elements, such as helices and loops, and calculates the thermodynamic stability of these individual components based on their sequence and structural context. The overall free energy of the RNA structure is then determined by summing the contributions from all these elements. This model relies on experimentally determined energy parameters from sources such as the Turner model [7] and the Andronescu model [8].
The nearest-neighbor model is based on the assumption that the energy of base pairs should not be considered in isolation, but in the context of their environment. Successive base pairs amplify their pairing energies by stacking, thereby increasing the overall thermodynamic stability of the secondary structure[3]. Using the energy parameters from the models mentioned above, one can calculate the energies of individual substructures within the RNA. These substructure energies are then combined to determine the total free energy of the secondary structure for the entire RNA sequence.

Minimum free energy (MFE) structures

Naturally, systems strive towards a state of minimum energy. The prediction of Minimum Free Energy (MFE) structures operates on the assumption that most naturally occurring RNAs adopt their most energetically favorable conformation. The first widely used algorithm for predicting MFE structures came from Nussinov et al. using a dynamic programming algorithm to predict the MFE structure of pseudoknot-free RNAs [9]. More recent tools such as MFold and RNAfold, rely on the energy models discussed in Section 1.2.1 as a foundation. Similar to Nussinov et al. they use dynamic programming algorithms to predict secondary structures efficiently, using calculated free energies to identify the most thermodynamically stable configurations. [10, 4, 11]

Partition function and ensemble free energy

While predicting the MFE structure provides valuable insights, it is important to recognize that RNA molecules can adopt multiple viable secondary structures. Various factors, such as kinetics and environmental conditions, can influence the structure an RNA molecule assumes in natural systems. To account for all possible states in the ensemble, the partition function is calculated:

$$Z = \sum_i e^{-\beta E_i} \tag{1.1}$$

- Z is the **partition function**, which represents the total statistical weight of all possible secondary structure (states) of the RNA molecule.

- $\sum_i$ denotes the **sum over all possible states** i of the RNA molecule. Each state i corresponds to a different possible secondary structure that the RNA can adopt.
- E_i is the **energy of configuration** i. This energy includes contributions from various interactions, such as base pairing, stacking, and loop entropy, which define the stability of each configuration.
- $\beta = \frac{1}{k_B T}$ is the **inverse thermal energy**, with k_B being the **Boltzmann constant** and T the **absolute temperature** of the system.
- $e^{-\beta E_i}$ is the **Boltzmann factor**, which determines the probability weight of each configuration i at a given temperature. Configurations with lower energy E_i have a higher statistical weight and are thus more likely to be occupied. For example the MFE structure which has the lowest energy structure in the ensemble has the highest probability of occurring in the ensemble.

Calculating the partition function for an RNA sequence opens the door to several other insights into the structural landscape and the statistics behind it. The probability of a given structure (i) in the ensemble is defined as :

$$P_i = \frac{e^{-\beta E_i}}{Z} \tag{1.2}$$

The **Ensemble Free Energy** corresponds to the Gibbs free energy of the ensemble, calculated by rearranging the terms in the partition function so that $F = -k_B T ln(Z)$. The calculation of the ensemble free energy serves mainly as a normalization factor for comparison with other ensembles. Which will be relevant in later sections regarding the design of RNA sequences, see section 3.3.2.[12]

By incorporating the statistics of the ensemble, thermodynamic models provide a more comprehensive view of the possible secondary structures that an RNA sequence may exhibit. Algorithms such as the one developed by McCaskill [13] use dynamic programming to compute the equilibrium partition function and base pair probabilities. Building upon this, the Maximum Expected Accuracy (MEA) structure prediction method emphasizes the likelihood of various base pair formations within RNA secondary structures. MEA identifies the structure with the highest cumulative probability of base pairs, aiming to construct the overall most probable structure. When combined with free energy calculations based on the nearest-neighbor model, MEA provides a powerful alternative to Minimum Free Energy (MFE) prediction methods[14].

In summary, thermodynamic models provide a fundamental framework for RNA secondary structure prediction by calculating the most stable configurations based on free energy principles. Structural prediction strategies which focus on Minimum Free Energy (MFE) or Maximum Expected Accuracy (MEA) use these models to identify likely structures, with the partition function extending this by capturing an ensemble of possible configurations. Through insights gained from the statistics of the ensemble, thermodynamic

approaches reveal a range of potential RNA structures, providing a comprehensive view of the structural ensemble at equilibrium.

However, while thermodynamic models capture the stability of RNA structures in a steady state, they cannot express non-equilibrium dynamics. They overlook the dynamic nature of RNA folding as it occurs in a cellular environment. To fully understand the functional forms of RNA, it's essential to consider the kinetics of folding, which involves time-dependent pathways and transient structures. The next section introduces kinetic models that address how RNA structure evolves in real time and how intermediate structures influence its final functional form.

1.2.2 RNA folding kinetics

While thermodynamic models offer insights into RNA's most stable structures, they do not account for the dynamic nature of RNA folding as it occurs in real-time. RNA folding is a kinetic process influenced by the rates of molecular configurations and sequence-specific pathways that determine how the RNA passes through intermediate structures to reach its final, functional conformation. Kinetic models try to capture and simulate those dynamic processes that contribute to the final structures of RNAs.
Before delving into kinetic models, it's important to acknowledge the different approaches used to study RNA folding dynamics. Molecular Dynamics (MD) simulations offer atomistic details by modeling the physical movements of atoms and molecules over time. While MD simulations provide high-resolution insights into the 3D structure of RNAs, they are computationally intensive and often impractical for modeling large RNA molecules or long folding timescales relevant in biological contexts. Therefore, this thesis focuses on coarse-grained kinetic models that capture essential folding dynamics with reduced computational complexity.[15]
This section focuses on the kinetics of RNA folding, with particular emphasis on how cotranscriptional folding affects the final secondary structure. Since this thesis focuses on cotranscriptional folding and introduces new tools that exploit this process, the chapter will primarily review software used to simulate cotranscriptional folding. Special attention will be given to DrTransformer, a coarse-grained kinetic simulator, as it exemplifies the kinetic simulation approach used by additional software introduced later in this thesis.

Transcription

Transcription is the process by which RNA is synthesized from a DNA template by an RNA polymerase. It is a three-step process that begins with **initiation**, where RNA polymerase recognizes and binds to a promoter sequence on the DNA. During **elongation**, RNA polymerase catalyzes the sequential addition of nucleotides complementary to the DNA template strand. This process elongates the RNA chain in a 5' to 3' direction, with each new nucleotide added to the 3' end of the growing RNA molecule. This movement creates opportunities for the nascent RNA to begin forming intramolecular base pairs, leading to the formation of secondary structures as the RNA folds cotranscriptionally. Such cotranscriptional folding can affect RNA structure and function by allowing certain regions

to fold before the entire RNA molecule has been synthesized, potentially affecting the final structure of the molecule. **Termination** occurs when RNA polymerase encounters a termination signal that causes the release of the newly synthesized RNA strand. The specific mechanisms of termination vary significantly across different organisms.[16]

Cotranscriptional folding

During the elongation phase that occurs after initiation, where nucleotides are added to the 3' end of the RNA molecule, nucleotides of the RNA can form base pairs with other nucleotides that have already been transcribed. This leads to the formation of intermediate secondary structures. Cotranscriptional folding plays a fundamental role in the functioning of life. For example, in certain bacteria, cotranscriptional folding can result in two different conformations, leading to either termination or continuation of transcription. The formation of a termination conformation can cause premature termination before any coding sequence is transcribed, whereas the formation of an antitermination hairpin prevents a termination factor from binding to the RNA, allowing the polymerase to continue transcription [17]. In the context of RNA folding, kinetic folding traps occur when intermediary structures form that result in a different conformation than the molecule's eventual minimum free energy (MFE) conformation. These traps arise during transcription, as the RNA chain folds sequentially. Stable intermediates, formed early in the transcription process, can persist due to their stability, potentially preventing the RNA from reaching its MFE structure immediately. This delay in transitioning to the MFE structure may affect the molecule's functional state, as the RNA may remain temporarily "trapped" in these metastable conformations until the full sequence is transcribed and conditions allow for further folding transitions [18].
Several factors influence the final secondary structure of an RNA molecule during cotranscriptional folding. Besides the nucleotide sequence itself and various proteins that may bind to the RNA, the rate of transcription plays a significant role. Transcription speed varies across organisms and is influenced by environmental conditions [19, 20]. Depending on the organism, transcription rates can range from 10 to 200 nucleotides per second [21]. Consequently, there is no single cotranscriptional folding path for a sequence, it depends on various factors that must be considered when simulating cotranscriptional folding.

Energy barriers and landscapes

The energy barrier is encountered when structure i transitions to structure j. This transitional path is defined as following:

Definition 3. *A **transitional** path $P_{i,j}$ between two secondary structures i and j is a sequence of secondary structures representing the transition from i to j. Each step in this sequence involves, at most, a single base-pair opening or closing event.*

An energy barrier represents the minimum amount of energy required to transition from one state to another. This energy barrier is defined as following:

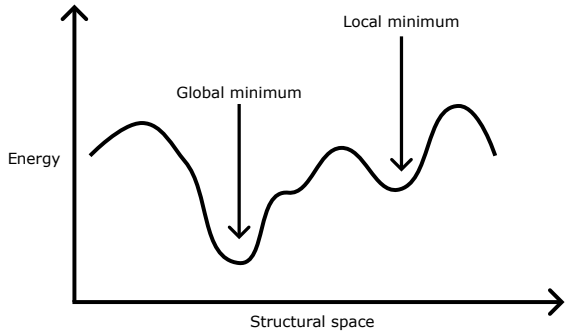


Figure 1.2: Visualization of the rugged energy landscape of a fictions RNA sequence.

Definition 4. *An **energy barrier** β_{ij} between two structures i and j is defined as the maximum free energy difference between structure i and the saddle point on the transitional path over all different possible paths.*

When there is a high barrier between two states, a significant amount of energy is required for the transition to occur. This means that the reaction is less likely to occur in some . In this work, these states correspond to different secondary structures of the same nucleotide sequence. The energy barrier therefore reflects the energy required to form or dissolve base pairs in order to transition to an alternative structure.
Typically, the system draws this energy from ambient temperature, meaning that the height of the barrier directly influences the probability of transitioning between states. By evaluating the energies of all possible secondary structures within the current ensemble, an energy landscape of these structures can be created. This landscape features a global minimum, corresponding to the Minimum Free Energy (MFE) structure, as well as multiple local minima. Often described as a rough and rugged energy landscape with many valleys and peaks [22]. While naturally occurring RNAs usually fold into the global minimum, they can sometimes become trapped in local minima, leading to misfolded structures that differ from the MFE structure.
During transcription, the energy landscape of an RNA molecule continuously shifts as new nucleotides are added to the growing sequence. This dynamic landscape makes it challenging for the RNA to consistently fold into its correct, functional conformation. Consequently, kinetic traps arise, which are intermediate structures that become stuck in local energy minima during transcription [23]. If these kinetic traps form and the RNA does not refold into the correct structure in time, misfolding can occur. As transcription progresses, the energy barriers between local minima may increase to a point where they are too high to overcome, effectively trapping the RNA in a misfolded state.

Kinetic modeling and simulation

The following three components are fundamental for kinetic modeling and the successive simulation. The **Conformation Space** includes all possible secondary structures that an RNA sequence can form, considering only Watson-Crick and GU wobble base pairs. The

Move set represents a set of elementary transitions allowed between conformations, such as the formation or dissociation of single base pairs[24, 25] or entire helices[26]. Simple moves (adding/removing base pairs) provide fine-grained resolution but are computationally demanding. Some models use helix-based moves, enabling faster exploration but introducing structural constraints. **Transition rates** between each nodes represent the rates for each transition between the nodes in the graph.
The conformational space and transition rates can be represented as a continuous-time Markov chain, expressed as:

$$\frac{dP_i(t)}{dt} = \sum_{j \neq i} (P_j(t)k_{ji} - P_i(t)k_{ij}) \tag{1.3}$$

Here $P_i(t)$ is the probability of observing conformation i at time t, and k_{ij} is the rate of transition from state i to j.

The transition rates between states can be deduced through various methods. One approach, as used in *Kinfold* [24], calculates the rate between two states i and j based on the free energy difference ΔG_{ij}. This free energy difference can be computed using the Turner energy model [7] and the kinetic rates can be acquired using a model introduced by Kawasaki [27]:

$$k_{ij} = \gamma e^{\frac{-\Delta G_{ij}}{2RT}} \tag{1.4}$$

Where γ corresponds to a factor that sets the timescale of the process.

Another method for determining kinetic rates involves using the Metropolis equation 1.5

$$k_{xy} = \gamma \cdot \max\left(1, e^{\frac{\Delta G(x) - \Delta G(y)}{RT}}\right) \tag{1.5}$$

Alternatively, kinetic rates can incorporate the activation energy between states using the Arrhenius equation1.6:

$$k_{xy} = \gamma e^{-\frac{\Delta G^{\ddagger} - G(x)}{k_B T}} \tag{1.6}$$

The arrhenius equation takes the activation energy between two states into account by adding $\Delta G^{\ddagger}$,which represents the free energy of the transition state (or the activation energy) between states x and y.

Establishing these three sets enables simulations of RNA folding kinetics, although under the assumption that the RNA's sequence length remains constant. This can be useful to simulate the dynamics of RNA secondary structures in solution or the folding process of a denatured RNA but in most cases the full length transcript of an RNA exists in a folded state due to cotranscriptional folding. To capture this dynamic process, kinetic models incorporate nucleotide addition events, creating a new model at each step that

simulates folding until the next nucleotide is added. Kinfold is an example of a simulator capable of modeling both folding of a fixed length sequence and cotranscriptional folding.

Kinfold is an example of a simulator capable of modeling both folding of a fixed-length sequence and cotranscriptional folding. It assumes that the basic steps in secondary structure formation are the opening and closing of base pairs. Kinfold utilizes these elementary steps to simulate RNA folding in a stochastic manner using the Gillespie algorithm. By applying the Gillespie algorithm, Kinfold selects each folding event based on calculated probabilities and assigns a time interval for each event to occur.
Using Kinfold for cotranscriptional folding, Kinfold recalculates the kinetic model with each nucleotide addition. After each addition, the system updates the ensemble of accessible structures and transition rates, modeling the folding dynamics until the next nucleotide arrives. This iterative process enables Kinfold to simulate how RNA folds in real time as it is synthesized, accounting for intermediate structures and potential kinetic traps, both of which significantly influence the RNA's final folded conformation.[24]

Coarsegrained models

Due to the computational expense of kinetic simulations, coarse-graining techniques can be employed to reduce computational costs. Coarse-graining reduces the number of states to be computed by allowing the dissociation of helices in singular steps [26] or by enabling transitions between different secondary structures per step [28].

The *DrTransformer* [29] algorithm applies coarse-graining to manage the complexity of the structural landscape after each nucleotide addition during the simulated transcription. The algorithm follows a four-step cycle, with each step corresponding to the extension of the sequence by a nucleotide:
(I) Extension and Landscape Expansion: In this initial step, DrTransformer expands the sequence by one nucleotide and generates an updated set of potential structural configurations. These configurations, or candidate structures, are derived from the ensemble of currently existing structures.
(II) Coarse-graining of Candidates: After generating candidate structures, DrTransformer applies a coarse-graining process to filter and condense the set of configurations. This step effectively reduces the number of potential structures, lowering computational complexity by focusing only on the most relevant folding pathways.
(III) Deterministic Simulation of Dynamics: With a reduced set of structures, DrTransformer performs a deterministic simulation to model the dynamics of the RNA folding process, recalculating and updating the occupancy probabilities for each candidate structure based on system changes.
(IV) Pruning of Low-Occupancy Structures: Finally, the algorithm removes structures with minimal or no occupancy from the ensemble. This pruning step further simplifies the structural landscape, ensuring that only meaningful conformations remain in the simulation.

In conclusion, kinetic simulations play a crucial role in understanding RNA folding dynamics, especially during cotranscriptional folding, where intermediate structures significantly influence the final conformation. Coarse-graining techniques, as demonstrated by tools such as DrTransformer, allow us to simulate these processes efficiently by reducing computational complexity without sacrificing essential details. Building on these concepts, this thesis introduces a domain-level cotranscriptional folding simulator based on the software Peppercorn[30], which will be introduced in chapter 3.

1.3 RNA design

Designing RNA sequences that fold into specific structures is also known as the inverse folding problem[31]. The process involves finding a nucleotide sequence that not only adopts a desired structure but also avoids folding into undesired alternative configurations. An aspect of this process is the balance between **positive** and **negative** design principles, which guide the selection and refinement of nucleotide sequences to favor certain structural outcomes while avoiding undesirable ones.
Positive design focuses on creating RNA sequences that favorably fold into the desired structure. In this approach, the sequence is tailored to maximize the stability of the target structure, often by minimizing its free energy. This method is particularly useful when a stable structure is essential for RNA function or binding specificity.
Negative design, on the other hand, aims to prevent the formation of alternative or incorrect structures. The goal is to minimize the stability of potential off-target configurations by increasing the specificity of the target structure. Often, both approaches are satisfied, as optimizing for both ultimately results in the most successful design. [32]
Designing RNA sequences imposes multiple challenges in finding a suitable nucleotide sequence that exhibits specific structural features. Recent findings have shown that the RNA design problem using the secondary structure prediction approach introduced by Zuker et al.[33], is NP-hard[34]. Therefore, it is beneficial to use heuristics for designing RNA sequences. The most common approaches begin by selecting a starting sequence and iteratively modifying it. Each iteration involves scoring the modified sequence based on an objective function to find the best-scoring sequence[31, 35].
There are three main components essential for successful RNA sequence design:

1. **Computational Model**: This defines how the structures of the final nucleotide sequence are calculated, including thermodynamic models like MFE and MEA, or kinetic models.

2. **Objective Function**: This function assigns a score to each resulting nucleotide sequence, allowing for the evaluation and comparison of candidate sequences based on how well they meet the design criteria.

3. **Optimization Algorithm**: This algorithm makes changes to the nucleotide sequence and reevaluates its fitness, guiding the search based on the objective

function toward optimal or near-optimal sequences

1.3.1 Computational model and objective function

As introduced in Section 1.2, several models for secondary structure prediction exist. Thermodynamic models like the Minimum Free Energy (MFE) structure focus on finding the most stable structure, making them ideal when the steadty-state structure is of primary interest. Alternatively, the Maximum Expected Accuracy (MEA) approach considers the overall highest probability over all base pairs, which can be useful for designs where ensemble behavior matters.
Kinetic models, on the other hand, simulate the folding pathways of RNA molecules and are essential when the folding process itself affects function, such as in co-transcriptional folding or riboswitch activation. The choice between thermodynamic and kinetic models depends on whether the design requires consideration of the final structure's stability or the dynamics of folding.
To find the best fitting nucleotide sequence, an objective function must be defined to score each resulting sequence, allowing effective comparison of candidate sequences based on key structural features. This objective function should represent all features of the intended RNA design to ensure a successful outcome. While thermodynamic parameters can be calculated efficiently, kinetic parameters require significantly more computational resources. Therefore, achieving an optimal balance between accurately representing the design objectives and managing computational costs is essential to an effective and feasible RNA design process.

1.3.2 Optimization algorithm

To identify RNA sequences with the highest fitness based on the objective function, an effective optimization algorithm is crucial. Most RNA design algorithms start with an initial sequence compatible with the design target and iteratively adjust the sequence, evaluating each version to determine its compatibility. Over time, various optimization and search algorithms have been introduced to iteratively modify nucleotide sequences to achieve desired characteristics. Below, two optimization methods are discussed.

- **Adaptive Walk**: One of the first approaches to RNA design, introduced by Hofacker et al.[31], used the adaptive walk to find the most fitting sequence. In an adaptive walk, the algorithm makes incremental changes to the sequence, evaluating each modification based on the objective function. If a change improves the fitness score, it is accepted, and the sequence is updated. If not, the change is discarded. While adaptive walks efficiently progress toward local optima, they often become "trapped" in these local solutions, unable to reach the global optimum. This limitation has led to the development of more advanced methods, such as simulated annealing, which can sometimes escape local optima by accepting suboptimal solutions.

- **Simulated Annealing**: To improve upon the more incremental nature of the adaptive walk, simulated annealing introduces a structured probabilistic approach. Inspired by the annealing process in metallurgy, this technique allows the algorithm to accept both beneficial and, occasionally, suboptimal changes in the sequence, helping to avoid local optima. As the algorithm progresses, it reduces the likelihood of accepting worse solutions, effectively "cooling down" the search space. [36]

1.4 Abstracting RNA

Abstraction layers simplify the structural complexity of RNA, making it easier to design and predict sequences without immediately addressing individual nucleotides. One such abstraction is the concept of RNA shapes, introduced by Griegerich et al. [37]. RNA shapes simplify structural information represented in dot-bracket notation into a more compact form by focusing on key structural features. At the highest level of abstraction, unpaired regions are omitted, emphasizing only the essential paired structures. Additionally, nested helices are merged into single units to further streamline the representation, allowing for a concise yet informative overview of RNA structure. For instance, the structure ..((..()..)). is abstracted as [()], focusing on the arrangement of helices. Using this abstraction level allows the focus to shift on the shapes of the RNA. Allowing predictions of RNA structures to be first made on an abstract level before continuing with the prediction on the nucleotide level.
Another approach to abstracting RNA, inspired by its widespread use in DNA nanotechnology[38, 39, 40], is the **domain level**. Unlike RNA shapes, which condense structural features into simplified shapes, the domain level focuses on grouping consecutive nucleotides into functional units called domains. A domain is defined as follows:

Definition 5. *A **domain** is a tuple $d = (r, \tau)$, where r is the name of the domain and τ its length. A domain is complementary to domains of the form $d^* = (r^*, \tau)$, their name has the suffix $*$ and their length is the same. By convention, it is forbidden that a system contains domains whose name indicates complementarity, but that have different lengths. A domain can only have one complementary domain: if $d = (r, \tau)$ and $(d^*)^* = ((r^*)^*, \tau)$, then $d = (d^*)^*$.*

Domains represent multiple consecutive nucleotides that can be considered as a single unit, either pairing with a specific complementary domain or remaining unpaired, allowing the focus to be on structural features and their specific lengths rather than on individual nucleotide interactions. The simplicity of the domain level eases the verification of sequence design before finding a suitable nucleotide sequence based on the domain-level sequence.

Similar to the dot-bracket annotation introduced in section 1.1, various annotations exist for secondary structures at the domain level. One annotation widely used in this thesis is the kernel-string annotation used by Peppercorn [30]. **Kernel-strings** combine the structural information of the dot-bracket annotation with the domain sequence,

allowing both structure and sequence to be represented in a single string. In this notation, an unpaired domain is represented solely by the domain name, while a paired domain consists of the domain name plus an open parenthesis. The corresponding binding domain is implied by the associated closing parenthesis, rather than being explicitly written. A key advantage of kernel-strings is their compact format, as they encapsulate both structural and sequence information in one cohesive string. However, it is important to note that kernel-strings are limited to representing pseudoknot-free structures, as the notation cannot capture the complex interactions present in pseudoknots.

Domain-level sequence:	a	a*	x	b	c	c*	b*	d	y	d*
Secondary structure:	(	)	.	(	(	)	)	(	.	)
Kernel notation:	a(	)	x	b(	c(	)	)	d(	y	)

2 Single stranded domain-level design

This chapter introduces a framework for designing domain-level sequences that fulfill an abstract cotranscriptional folding path (aCFP). An aCFP describes the sequential folding and refolding of helices during transcription without specifying their lengths or nucleotide sequences explicitly. The main idea is using domains as an abstraction layer to simplify sequence design while enabling control over cotranscriptional folding dynamics. By breaking down cotranscriptional folding at the domain-level into a defined set of rules, the framework enables the design of domain-level sequences that follow a specific cotranscriptional folding path. These rules govern key processes found in cotranscriptional folding such as the extension of the sequence with new domains, the establishment of pairings, and the refolding of structures via branch migration events. An essential step in the framework is verifying traversability. A traversable aCFP indicates that transitions between the steps of the aCFP are feasible, i.e. there exists a sequence of rules whose application leads to the target aCFP. Key mechanisms for controlling these transitions, include the use of short domains in the form of toeholds to initiate structural transitions and clamps to prevent undesired reactions. Using the established rule set together with information derived from the aCFP, it is possible to determine whether the aCFP can be translated into a domain-level sequence that satisfies the pairing constraints defined by the aCFP. Such an aCFP is termed sls-translatable. However, due to the inherent complexity of the design problem, the algorithm presented at the end of this chapter supports only a subset of all sls-translatable aCFPs. This subset is called coco-translatable and represents the class of aCFPs that can be verified and realised by the current implementation of the sls-design algorithm provided by the software CocoPath (see chapter 3).

2.1 Cotranscriptional folding at the domain-level

Domains are not only a convenient abstraction for nucleic acid structure, but they are also a suitable abstraction layer for analyzing possible conformational rearrangements. Consistent with existing models for domain-level strand displacement systems [30], four basic reaction types are distinguished at the domain-level: **bind**, **unbind**, **three-way branch migration** and **four-way branch migration**. Section 3.2 discusses the development of a domain-level simulator that is based on the existing software peppercorn [30] to enumerate all valid domain-level cotranscriptional folding paths. In this section, we establish a rate-independent model of cotranscriptional folding that can help decide whether unimolecular rearrangements during cotranscriptional folding pathways result in a particular structure. Intuitively, we distinguish four events. **Extension:** a new domain gets added to the 3' end. Initially, a domain is always treated as unpaired. **Folding:** A

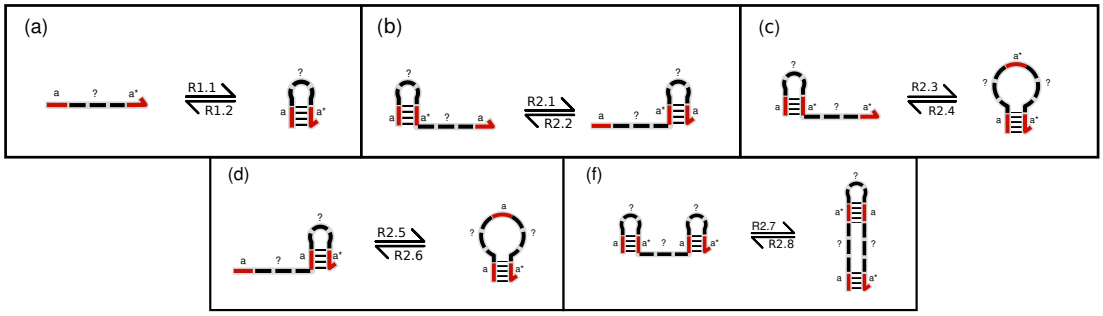


Figure 2.1: Visualization of all folding and refolding rules including unbinding. Red segments represent one or more subsequent domains that engage in a reaction as one unit. The dashed lines represent wild-card regions ?. Note that these regions must be valid (or "well-formed") secondary structures, otherwise the red domains cannot interact in a pseudoknot-free model.

domain pairs with an unpaired complementary domain. This step is considered instantaneous if it yields a valid secondary structure. We guarantee that ambiguity at this step can always be resolved through subsequent refolding. **Unfolding**: Two previously paired domains become unpaired. **Refolding:** A domain changes its existing pairing status through three-way or four-way branch migration. Depending on the transcription speed and the speed at which the refolding event occurs, only a limited number of refolding events are possible before a new expansion event occurs.

2.1.1 Single-stranded domain-level folding rules

A set of rewrite rules is provided below to enumerate all possible domain-level reactions. Each rule is represented as a transformation on a kernel string and can be applied when the reactant pattern matches a substring of the kernel-string, resulting in a corresponding product kernel-string. The symbols used in the rules are defined as follows: `a` denotes a segment of consecutive domains of arbitrary length, and `a*` represents the complement of that segment; the wildcard symbol **?** denotes an arbitrary, valid secondary structure. All rules obey mass conservation: the number of symbols (`a`, `a*`, and **?**) is the same on both sides of each rule, although `a` and `a*` may be implicit in the kernel notation.

- Rule 1.1 (`R1.1`): `a ?  a*` → `a( ?  )`
- Rule 1.2 (`R1.2`): `a( ?  )` → `a ?  a*`
- Rule 2.1 (`R2.1`): `a( ?  )  ?  a` → `a ?  a*( ?  )`
- Rule 2.2 (`R2.2`): `a ?  a*( ?  )` → `a( ?  )  ?  a`
- Rule 2.3 (`R2.3`): `a( ?  )  ?  a*` → `a( ?  a* ?  )`
- Rule 2.4 (`R2.4`): `a( ?  a* ?  )` → `a( ?  )  ?  a*`

- Rule 2.5 (R2.5): `a ?  a( ?  ) → a( ?  a ?  )`
- Rule 2.6 (R2.6): `a( ?  a ?  ) → a ?  a( ?  )`
- Rule 2.7 (R2.7): `a( ?  )  ?  a( ?  ) → a( ?  a*( ?  )  ?  )`
- Rule 2.8 (R2.8): `a( ?  a*( ?  )  ?  ) → a( ?  )  ?  a( ?  )`

The rules (see also Fig. 2.1) denote the following events: R1.1 and R1.2 represents folding and unfolding, where two initially unpaired complementary domains pair or unpair. R2.1 - R2.6 provide an exhaustive list of three-way branch migration reaction patterns. R2.7 and R2.8 express all possible four-way branch migration reactions. The inherent symmetry of four-way branch migration allows us to express all contexts with only one reversible rule.

2.1.2 A rate-independent model of cotranscriptional folding

Using the rule set outlined in the previous section, a rate-independent model of cotranscriptional folding can be established. To achieve this, a new rule, denoted as **RX**, must be first introduced. This rule represents the extension of a sequence by one domain and is defined as follows:

- Rule X (RX): `? → ?  a`

Cotranscriptional folding pathways are modeled here by applying rules based on their rates. The rules are divided into four different categories, instant, fast, slow and negligible. R1.1, is categorized as instant, meaning that if R1.1 is applicable, it must be applied immediately. [R2.1, R2.2,...,R2.8] are classified as fast. While RX, is slow and R1.2, is considered negligible, as unpairing is not included in this model due to its kinetic disadvantage.
These categories approximate kinetic rates, capturing the approximation that in this model, binding between two unpaired domains occurs first, with three-way or four-way branch migrations following once all pairings have been established. After all structures are found using [R2.1, R2.2,...,R2.8], RX can be applied to extend the system. R1.2 will never apply in this model. This leads to structures where the only point where unpaired complementary domains can exist is the time frame between RX and R1.1.
Later in this thesis section 3.2 will introduce a rate-dependent model of cotranscriptional folding for domain-level sequences.

2.1.3 Controlling structural rearrangements using refolding rules

The design uses three domain types to ensure correct folding according to the rules in section 2.1.1. Wether folding is indeed efficient depends on kinetic simulations, where domain lengths are explicitly taken into account. To achieve correct folding, the design framework uses three distinct domain types, each serving a specific function.

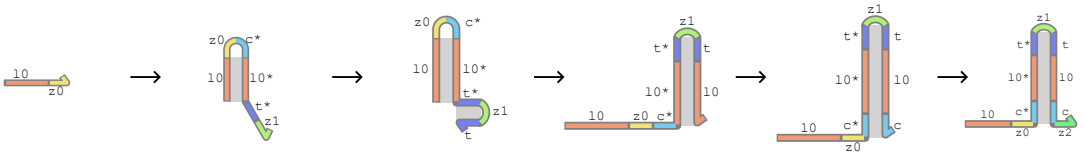


Figure 2.2: Shows the cotranscriptional folding path of a domain-level sequence. At each step a domain gets added to the sequence and the thermodynamically most stable structure is displayed.

1. **Long domains** (l) function as logical entities in the sense that their binding must be consistent with the structural constraints imposed by an abstract cotranscriptional folding path (aCFP). Long domains can change their binding status from bound to unbound by three-way branch migration or exchange binding partners by four-way branch migration.

2. **Short domains** (s) serve as **toeholds** (t) or **clamps** (c), depending on context. Toeholds facilitate branch migration by reducing the kinetic barrier, whereas clamps inhibit branch migration by raising the barrier.

3. **Timer domains** (z) provide the necessary time for upstream folding to complete before the next aCFP step initiates. There are no complementary domains to timer domains in the system, such that they are generally assumed to be unstructured. By controlling the length of the timer domain, one can vary the time it takes to transcribe the domain and thereby delay the transcription of downstream domains. For notational convenience, the absence of a timer domain can be specified by specifying as a timer domain of the length $\tau = 0$.

The addition of short domains as toeholds and clamps can significantly influence the outcome of previously uncertain reactions. Similar to toehold-mediated strand displacement, toeholds are used to initiate branch migration by lowering the kinetic barrier [41]. In contrast, clamps are used to raise the kinetic barrier for certain strand invasions, ensuring that the reverse reaction is only favorable and achievable when intended. An example of how the toehold initiates branch migration is shown in Fig. 2.2. In the design scheme proposed in this thesis, toeholds and clamps are used to regulate the pairing partners of a long domain, thereby controlling the refolding events during transcription. This design is called the **sls-scheme**, representing a short domain followed by a long domain and ending with another short domain. By incorporating toeholds and clamps, the scheme ensures precise regulation of pairing partners during transcription. Using short domains to either lower or raise the kinetic barrier, we can define a target conformation, which is thermodynamically favored and kinetically accessible. Below is the applied domain rule set denoted as **dR**, with short domains added to facilitate reactions, also shown in Fig. 2.3.

- dR1: `L ? L* → L( ? )`
- dR2.1: `L( ? c* ) t* ? t L c → L ? c*( L*( t*( ? ) ) )`
- dR2.2: `c L t ? t* L*( c* ? ) → c( L( t( ? ) ) ) ? L`
- dR2.3: `t L( c ? ) ? c* L* t* → t( L( c( ? L* ? ) ) )`
- dR2.4: `c L( t ? t* a* L* ? ) → c( L( t( ? ) ) ) ? L*`
- dR2.5: `L( ? c L t ? t* ) c* → L ? c( L( t( ? ) ) )`
- dR2.6: `t L c ? L( ? c* ) t* → t( L( c( ? L ? ) ) )`
- dR2.7: `t L( c ? ) ? L( ? c* ) t* → t( L( c( ? L*( ? ) ? ) ) )`
- dR2.8: `L( ? L*( ? c ) t ? t* ) c* → L( ? ) ? c( L( t( ? ) ) )`

In contrast to the abstract rule set, where reactions and products in rules `R2` are equally favorable, this applied rule set strictly enforces the formation of thermodynamically and kinetically favored structures. The following sections will use this rule set to guarantee a correct generation of domain-level sequences, that satisfy a specific aCFP.

2.2 Abstracting RNA for single stranded domain-level design

The design framework presented in this chapter uses three levels of abstraction to represent RNA sequences and their corresponding secondary structures. It begins with the specification layer, representing the most abstract level of the RNA molecule, which can be abstracted as an RNA shapes representation(see sec. 1.4), where only the order of the helices and unpaired regions in the sequence are specified. This is followed by the domain-level, which provides an intermediate representation that bridges the specification layer and the nucleotide layer. Each layer offers different levels of detail about the RNA molecule.
Figure 2.4 illustrates the sequence and folding path at the different abstraction levels, as well as the process of transitioning between them. The domain-level is accessible through the application of a translation algorithm, described in section 2.4. Verifying whether a domain-level translation was successful involves checking if the long domains are paired as intended by the aCFP. To focus only on the relevant interactions, we introduce the **Long-Domain Pairing Notation (LDP Notation)**, which captures only the pairings of long domains while omitting short and timer domains. This notation reduces the structure to its essential long domain pairings, providing a simplified representation corresponding to an aCFP that highlights whether the intended interactions are correctly formed. Section 3.3 introduces an algorithm that takes information from the domain-level and generates a nucleotide sequence that satisfies the pairing constraints from the specification level.

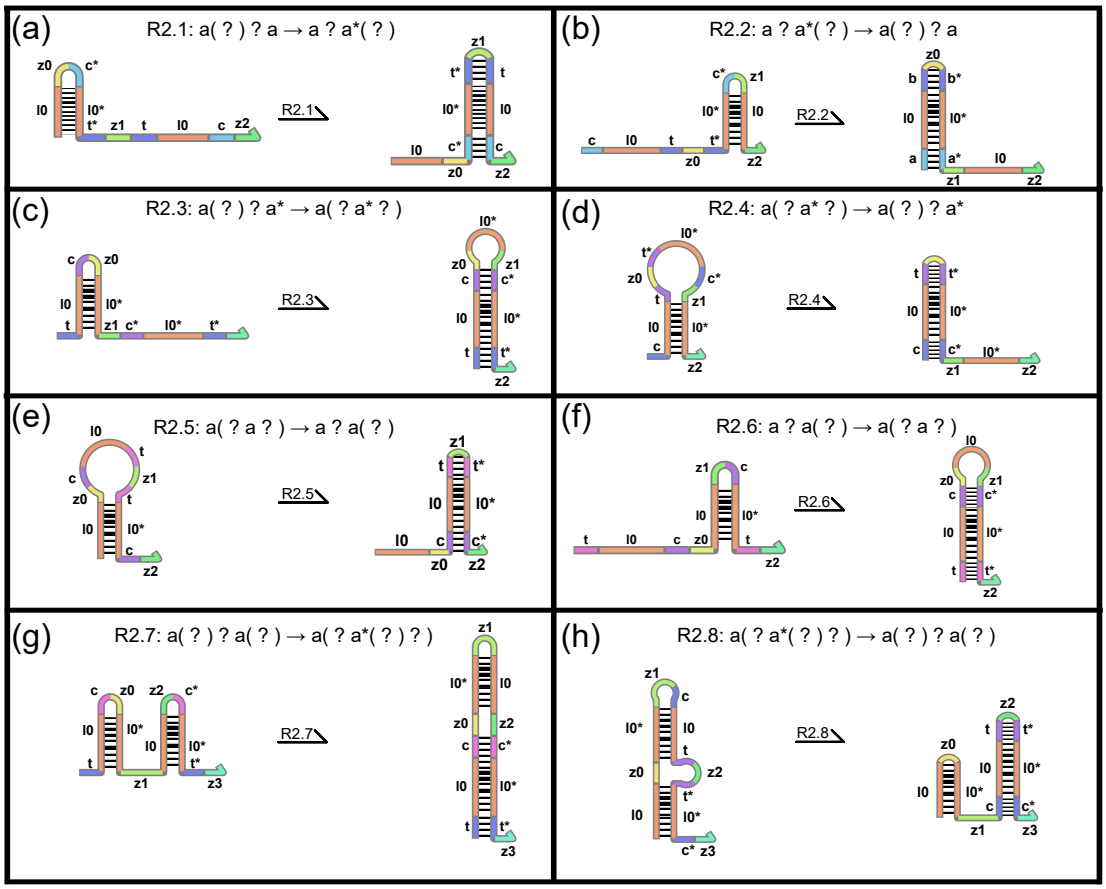


Figure 2.3: Shows the applied rule set for possible structural rearrangements. Short domains in reactant configuration are shown as unpaired for visual correspondence to the rules in figure 2.1. `l0` denotes the long domains, `z` denotes the timer domains, `t` denotes toehold domains and `c` denotes clamp domains. A pairing between clamp domains on the left side of the reaction would result in pseudoknot, while the toeholds are always able to form.

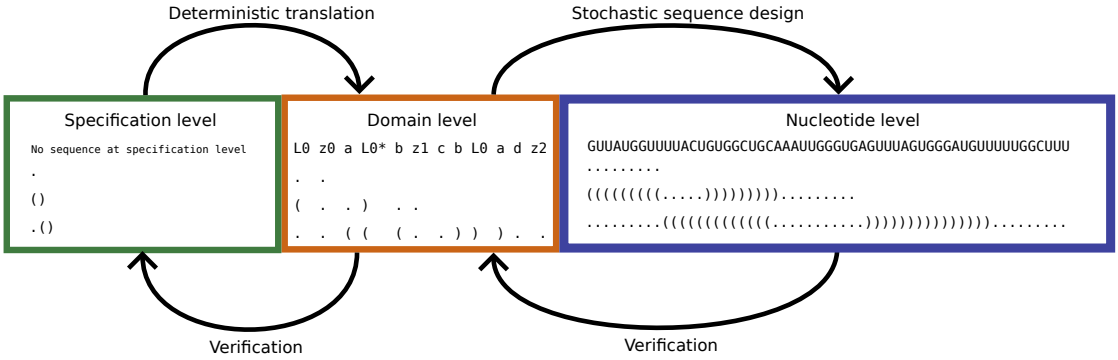


Figure 2.4: This figure shows the different abstraction layers for an sls-translatable aCFP. Using a deterministic translation scheme described in this section yields the domain-level sequence and consequently the domain-level folding path. The stochastic sequence design introduced in section 3.3 generates a nucleotide level sequence based on the information gathered on the domain-level. Verification across layers utilizes the Long-Domain Pairing (LDP) Notation (see sec. 2.2) to verify that the long domains are correctly paired according to the aCFP

Specification Level: The advantage of the specification level in designing folding paths lies in its ability to focus exclusively on large-scale structural transitions, bypassing the need to consider the specific sequence constraints or helix lengths. This simplifies the design process, allowing attention to be focused on overall structural dynamics rather than detailed sequence requirements. On the specification level RNA is only abstracted as a pairing constraint, indicating the order of helices in the structure. A pairing constraint is defined as follows:

Definition 6. *A **pairing constraint** is a set of index tuples (i, j) subject to the following conditions: (1) pairs are consistent: if $(i, j), (i, k) \in x$ then $j = k$, (2) pairs are nested: if (i, j) and (p, q) and $i < p < j$ then $i < q < j$.*

Pairing constraints can be represented using a dot-bracket notation [4] with following alphabet $A = \{., (,)\}$, where parenthesis indicate that the corresponding positions must be paired and a dot represent an unpaired position.
Pairing constraints at the specification layer define the order in which helices occur in the RNA sequence, but do not consider the length of those helices. Using the specification layer allows the design of complex structures and cotranscriptional folding paths. Pairing constraints can be used to define an abstract version of a folding path, showing the structural changes over time.

Definition 7. *An **abstract cotranscriptional folding path (aCFP)** is a list $C = C_1, C_2, \ldots, C_n$, where each C_i is a string notation of a pairing constraint, such that length of C_i is i.*

The specification level alone does not provide a means to define sequences that adhere to any given aCFP. The domain-level serves as the first abstraction layer where a logical framework can be established to ensure sequences conform to an aCFP. At this level, domain-level sequences follow the principle that, at any given time, the most favorable structure is the one in which the number of paired domains is maximized.

Definition 8. *A **maximum paired domain structure** (MPD-structure) is the structure of a domain-level sequence that maximizes the number of paired domains.*

Although some domain sequences could theoretically support multiple MPD structures, in practice the algorithm produces sequences with only one unique MPD structure.

Definition 9. *A domain-level sequence **satisfies** a pairing constraint if the LDP-notation of the MPD-structure corresponds to the pairing constraint.*

The domain sequence is partitioned into discrete units called, **segments**, such that the index i of a pairing constraint C_i corresponds to the number of transcribed segments required to satisfy that constraint.

Definition 10. *A domain-level sequence **satisfies** an aCFP, if, for every pairing constraint C_i in the aCFP, the partial domain sequence consisting of the first i segments, satisfies the pairing constraint C_i.*

The domain-level, allows a formal, logic evaluation, and approximate kinetic simulations after the inclusion of length specifications in the sequence. Upon approximate kinetic verification, the nucleotide level allows verification of the designs using established tools [24, 29].

2.2.1 Segments as building blocks for domain sequences

A segment is implemented as a sequence of domains that contains exactly one long domain and ends with a timer domain. Optionally short domains can flank the long domain on either side as needed. To maintain symmetry, the number of short domains on either side of the long domain must be equal.

The i^{th} position of each constraint corresponds to the i^{th} segment and each new constraint in the aCFP adds a segment, affecting the sequence's structure. The number of segments therefore correspond to the length of the aCFP. The purpose of segments is to ensure that the pairing constraints are met by allowing the long domains to form the necessary pairings for the current step. At the same time, the short domains facilitate refolding, guiding the structure toward one that is both thermodynamically favored and kinetically accessible, as specified by the pairing constraint. The timer domains function as a temporal buffer to allow intended upstream folding events to complete before the next segment gets transcribed.

Claim 1. *Any competition between the long domains of segments, can be resolved by appending complementary short domains to the segments paired in the target conformation.*

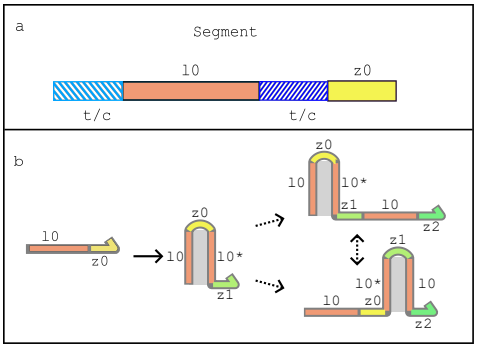


Figure 2.5: a)Shows the layout of a segment. Long domain($l0$) and timer domain($z0$) are present in all segments while short domains acting as toehold(t) or clamps(c) are added depending on the context. b) Displays the possible structure after each segment has been transcribed for a 3 segment long sequence without any short domains. Resulting in a system where no clear structure is dominant at the end of transcription.

Sequences composed of multiple segments fail to achieve a well-defined dominant structure in the absence of short domains, as shown in figure 2.5. To overcome this, the inclusion of at least one pair of short domains ensures the emergence of a dominant structure. If a competition is balanced, introducing a single pair of short domains is sufficient to drive the system toward the target structure. Specifically, one domain functions as a toehold, initiating the refolding process, while the clamp stabilizes the newly formed structure and prevents backward transitions. By increasing the shared domains between segments, the affinity between these segments rises, favoring the structure where the added domains form stable pairings, corresponding to the maximum paired domain structure (MPD-structure). These observations validate claim 1, demonstrating that the introduction of short domains is a necessary and effective mechanism to ensure the formation of dominant and stable structures.

Moving from one pairing constraint to the next represents a refolding process between two structures as a new segment is added. Because transcription continuously adds domains, there is limited time before the next refolding occurs. Thus, each refolding must complete before the next segment is appended, ensuring the structure is finalized after the timer domain is transcribed. By adjusting the length of these timer domains, the available time for structural refolding can be artificially extended.

2.3 Designing abstract cotranscriptional folding paths (aCFPs)

This thesis formulates the sequence design problem as a series of structural constraints that must be satisfied during transcription. Here, sequence design is approached as the

translation of these structural constraints into a domain-level sequence. These constraints are specified as an aCFP(see def. 7).

Definition 11. *A domain-level sequence design δ is **successful** if, for each pairing constraint C_i in the aCFP, the MPD-structure of the partial domain sequence δ_i comprising the first i transcribed segments satisfies the pairing constraint C_i and occupies more than* 50% *of the cotranscriptional structural ensemble.*

Successful translation of an aCFP into a domain-sequence means that the target structure maximizes the number of pairs at each partial sequence. The possible pairing constraints for length one, two and three are the sets {.}, {.., ()}, and {..., ()., (.), .()}, respectively. The Cartesian product of these sets gives all possible aCFPs of length three. While all possible aCFPs of length three can be realized at the domain-level using the sls-design, this is not the case for sequences of length four and beyond. An aCFP that can result in a successful domain-level sequence is termed **sls-translatable**.
Due to the complexity of the design problem, a complete and general definition of sls-translatability cannot be provided. The next section defines a subset of aCFPs, referred to as **coco-translatable**, which can be successfully translated using the current algorithmic implementation of the sls-design scheme. This subset represents the space of pairing constraint paths verifiable and realizable by the software **CocoPath**, introduced in chapter 3.

2.3.1 Checking translatability of an aCFP

To come up with sequences that fold and refold as specified by an aCFP, the respective pairing constraints are translated into domain-level sequences with specific folding and refolding behaviors. This work defines an aCFP as coco-translatable if it meets three criteria: (i) **assignability**, (ii) **consistency** and (iii) **traversability**. These three criteria represent the fundamental requirements for a successful translation:

- Assignability ensures that a valid domain-level sequence exists which satisfies all pairing constraints.
- Consistency is a thermodynamically motivated criterion that ensures the existence of a domain-level sequence in which, at each step, the MPD-structure (i.e., the structure with maximum domain pairing) of that partial sequence corresponds to the target structure.
- Traversability is a kinetically motivated criterion that requires each transition between pairing constraints to be achievable using the defined set of folding rules.

Together, these criteria define the space of coco-translatable aCFPs. Each criterion, together with its verification method in CocoPath, is detailed below.

Definition 12. *An aCFP is **assignable** if there exists a domain-level sequence δ_n compatible with each pairing constraint.*

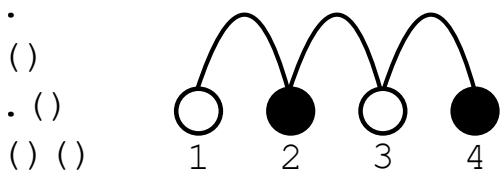


Figure 2.6: Visualization of a graph created to check if the aCFP on the left is assignable. Each character in the aCFP corresponds to a node in the graph. This visualization demonstrates an assignable aCFP, as confirmed by the bipartite graph.

Assignability is the first and most fundamental criterion that must be verified. It ensures that at least one domain-level sequence exists that is compatible with all competing secondary structures. For example, the aCFP $[.,..,...]$ is trivial to satisfy by assigning a sequence in which no complementarity exists between domains. However, as more complex pairings appear within the aCFP, dependencies emerge, introducing constraints where complementarities must be preserved in domain assignments.

Assignability can be verified by constructing a graph similar to multistable sequence design [35] see also figure 2.6. This graph is referred to here as an **aCFP-graph**. Vertex v_i corresponds to the i^{th} character in each pairing constraint. An edge is introduced for each assigned pairing constraint between the corresponding vertices. As secondary structures permit pairings only between complementary positions assignability of the aCFP is determined by verifying whether the resulting graph is bipartite, which confirms that all pairings are valid under the given constraints. If the graph consists of multiple disconnected subgraphs then vertices in different subgraphs will never form pairings with each other. Thus, assigning the same domain to vertices in different subgraphs is unnecessary and increases the risk of unwanted side interactions. Based on the graph, an array is constructed with separate entries for each vertex, where each entry takes the form X and X^*. Here, X is an integer differentiating between the subgraphs, while X^* denotes its complementary domain.

Definition 13. *An aCFP is* ***consistent*** *if the corresponding aCFP-graph, constructed during assignability, satisfies the following two conditions:*

- *The aCFP-graph contains no cycles.*
- *It is possible to assign a weight to each edge in the graph such that higher weights correspond to higher binding priority. These weights must be assigned such that, when examining the subgraph formed by the nodes* 1 *through* i *(for each* $i \leq len(aCFP)$*), corresponding to the pairing constraint* C_i*, satisfies the following: the sum of the weights of the active edges (i.e., those presenting the base-pairs in* C_i*) is strictly greater than the total weight of any alternative valid pairing within the same subgraph.*

Intuitively, this definition ensures that each target structure in the folding path is the uniquely most favorable conformation at the point it appears. The existence of a consistent weight assignment enforces a priority order in which domain-level interactions are formed. This order guarantees that competitions between structures can be resolved unambiguously during transcription, and that sequences can be designed to follow this order without conflicting pairing preferences.
A simple example of an aCFP that is assignable but not consistent is given by the path

$$[\,. \to () \to .() \to ()() \to ().()]$$

In this case, C_3 and C_5 introduce a conflict in pairing priority: the domain-level structures `().` and `.()` compete for the same region, and no priority can be assigned that resolves this conflict in favor of both target conformations.
Another example of inconsistency arises in the path

$$[\,. \to () \to .() \to (()) \to (.())].$$

Here, the final structure $C_5 =$ `(.())` introduces a cycle in the aCFP-graph. In contrast, the alternative path with $C_5 =$ `(().)` would avoid this conflict and the resulting aCFP graph would remain consistent.

Checking for cycles in the graph is straightforward, while the edge weighting is carried out as follows: Initially, each edge is assigned a weight of 0. The weight assignment works by comparing **active edges** (base pairs present in the current pairing constraint) with **inactive edges** (possible but unused pairings). If an inactive edge shares a vertex with an active edge, it is considered a competitor. To ensure consistency, each active edge must have a strictly higher weight than any competing inactive edge.
Weights are assigned step by step by iterating through each pairing constraint in the aCFP. At each step, unassigned active edges are given weights such that they have the highest priority among all competing edges present at that point. Once a weight has been assigned to an edge, it remains fixed and cannot be changed. After the full weight assignment is complete, each subgraph is checked to verify that the target structure C_i matches the structure with the highest possible total weight. This is done using a modified version of the Nussinov algorithm [9], adapted for domain-level interactions, where candidate base pairs are scored according to their assigned edge weights. See algorithm 1 below for the complete pseudocode.

Input: $aCFP$
Output: Boolean indicating whether given aCFP is consistent
$W \leftarrow$ empty mapping ; `// Mapping from edges to assigned weights`
for *each node* v *in* $aCFP$ **do**
  v.max_weight $\leftarrow 0$
end

// **Phase 1: Weight assignment**
for $i \leftarrow 0$ **to** len$(aCFP) - 1$ **do**
  $G_i \leftarrow$ getSubgraph$(aCFP, i)$; `// Subgraph corresponding to constraint` C_i
  for *active edge* e *in* G_i *that is unassigned* **do**
    $l \leftarrow$ getNode$(e[0])$; `// Left node of edge` e
    $r \leftarrow$ getNode$(e[1])$; `// Right node of edge` e
    $edge\_weight \leftarrow \max(l.\text{max\_weight}, r.\text{max\_weight}) + 1$
    l.max_weight $\leftarrow edge\_weight$
    r.max_weight $\leftarrow edge\_weight$
    $W[e] \leftarrow edge\_weight$
  end
end

// **Phase 2: Verification Using Modified Nussinov Algorithm**
for $i \leftarrow 0$ **to** len$(aCFP) - 1$ **do**
  $G_i \leftarrow$ getSubgraph$(aCFP, i)$
  $S_i \leftarrow$ ModifiedNussinov(G_i, W) ; `// Compute MPD-structure`
  if $S_i \neq C_i$ **then**
    return False ; `// Mismatch found at step` i
  end
end
return True

Algorithm 1: Pseudocode for consistency check

Definition 14. *An aCFP is* ***traversable*** *if it is assignable to some sequence* δ_n*, such that for each transition from* C_i *to* C_{i+1} *in the aCFP there exists a sequence of rules* $[R_{1.1}, R_{2.1}, \ldots, R_{2.8}]$*, applied according to their priority, that allows the transition.*

Checking the traversability of an aCFP corresponds to verifying whether the folding path is kinetically feasible. Importantly, this check does not consider the overall path as a whole, but rather evaluates whether each individual transition between consecutive constraints is achievable using the previously defined rules. For example, the path $[., (), .()]$ is traversable since it requires $R_{1.1}$ for the transition from $C_1 \rightarrow C_2$, $R_{2.1}$ for the transition from $C_2 \rightarrow C_3$. In contrast, the path $[., (), ...]$ is not traversable, as it would require unbinding, which is not permitted under the current traversability definition. Another example, the path $[., (), .(), .()., .(())]$, is also not traversable because the transition from $C_2 \rightarrow C_3$ would require the application of $R_{1.1}$ leading to the structure $(())$ in C_3. Note that due to the rule priority(sec. 2.1.2) $R_{1.1}$ always applies first if applicable. This results

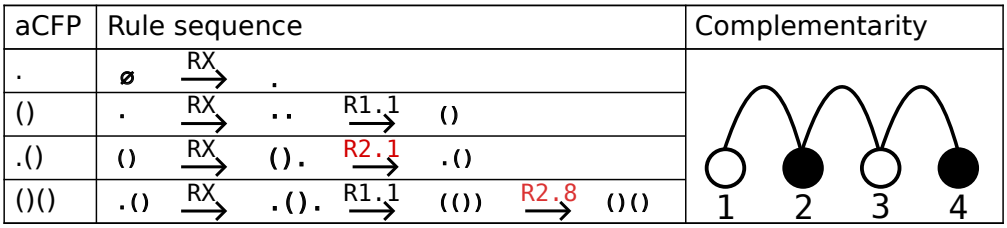


Figure 2.7: An example illustrating the deconstruction of an aCFP into a sequence of rules is shown. For clarity, only dot-bracket notation is used to represent structures. The complementarity graph, generated during the assignability check, is included to show the complementarity. The first refolding rules applied are highlighted in red.

in all structures being fully saturated by the end of the traversability check, ensuring that no constraint contains two unpaired, complementary domains remaining unbound. The traversability check is done by checking whether or not there is a sequence of rules that can reconstruct the aCFP.

Determining the rule sequence for an aCFP

The derivation of an aCFP into a sequence of rules involves going sequentially through each pairing constraint in the aCFP. The process begins with an empty structure and proceeds through three steps for each structural constraint in the aCFP. At every step, the complementary relationships established during the assignability check must guide the application of rules to ensure compatibility with previous constraints. Throughout this process, the sequence of rules is stored in an array, each rule corresponding to one entry (see also Figure 2.7):

1. Extend the structure by applying `RX`.
2. Apply `R1.1` if applicable.
3. Apply as many rules from `R2.X` as necessary to satisfy the structural constraint for the given length. Each rule, except the first after `RX` or `R1.1`, must affect a character previously paired or unpaired by the preceding rule. If no sequence of `R2.X` rules achieves the constraint, the target structure cannot be designed using the design scheme.

Checking traversability verifies that the transitions between successive constraints in the aCFP are achievable based on the logical framework described in this chapter. Often multiple rule sequences can achieve the same transition. Finding any valid sequence confirms that the aCFP is traversable. Once traversability is confirmed and the

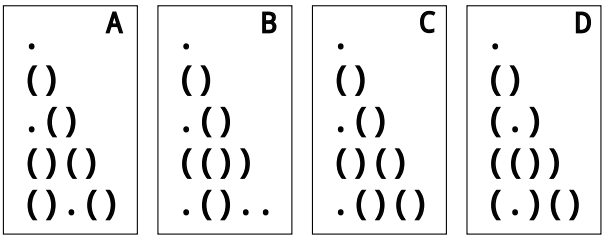


Figure 2.8: Figure A-D show 4 different folding pathways. Can you determine which one is sls-translatable? The answer is given below: (A) The aCFP is assignable, traversable, but not consistent. (B) The aCFP is assignable, consistent, but not traversable. (C) The aCFP is assignable, traversable and consistent. (D) The aCFP is not assignable.

corresponding rule sequence is determined, the design of a domain-level sequence can begin.

2.3.2 Check your understanding

Fig. 2.8 shows four folding paths. Can you determine which of these aCFPs are sls-translatable?

- `A`: The aCFP is inconsistent because the structure ".()" is defined at indices (0-2) in step 3, but a different substructure "()." appears at the same indices in step 5
- `B`: This aCFP is consistent and assignable but not traversable. The folding path is valid until step 4. The transition `(())` → `.()..` cannot occur because unbinding is not allowed when checking for traversability.
- `C`: Passes all checks and is therefore ready for translation.
- `D`: In D, index 1 pairs with indices 2, 3, and 4, requiring these indices to be complementary to 1. However, step 4 specifies that indices 2 and 3 must pair with each other, creating a conflict. As a result, no domain-level sequence can satisfy all pairing constraints.

Section 4.1 explores the combinatorial space of aCFPs by examining the sizes of the sets associated with different combinations of designability checks. This analysis is important because it shows which constraints are most restrictive and which are more easily satisfied, providing insight into the inherent challenges of achieving fully translatable folding paths.

2.4 Algorithm for domain-level sequence design

The algorithm proposed in this thesis for translating an aCFP into a domain-level sequence consists of two main stages:

1. **Translatability check:** This stage determines whether the given aCFP is coco-translatable. It evaluates assignability, consistency and traversability, providing crucial information for the subsequent steps. See section 2.3.1 for detailed information on how translatability of an aCFP is checked.

2. **Sequence generation:** Using the information obtained from the translatability check, this stage generates sequences through three sub-tasks:
   - **Assignment:** Assign initial long domains and their complementary sequences, e.g., $[L_0, L_0^*, L_0, L_0^*]$.
   - For each length i, extend the initially assigned long domains with short domains in a two step process:
     - **Balancing[i]:** Ensure that all competing structures at step i are at least equally favorable by balancing the number of domains.
     - **Triggering[i]:** Ensure that the target structure is favored among competing structures.

Input: $aCFP$: Array of constraints
Output: S: Domain sequence
$A \leftarrow$ assign_longdomains(aCFP) ; `// Assign initial long domains`
$S \leftarrow [\,]$; `// Initialize empty domain assignment list`
$x \leftarrow 0$; `// Tracks the index for unique short domains, denoted as` d_x
for $i \leftarrow 0$ **to** $len(aCFP) - 1$ **do**
  Append $A[i]$ to S;
  $pt \leftarrow$ convert_pt($aCFP[i]$) ; `// Convert constraint to pairtable`
  if $pt[i] \neq -1$ **then**
    $S[i] \leftarrow$ complement($S[pt[i]]$) ; `// balancing[i]`
    $S[i] \leftarrow [d_{x+1}, S[i], d_x]$; `// triggering[i]`
    $S[pt[i]] \leftarrow [d_x^*, S[pt[i]], d_{x+1}^*]$;
    $x \leftarrow x + 2$;
  end
end
for $j \leftarrow 0$ **to** $len(S) - 1$ **do**
  Append z_j to $S[j]$; `// Add timer domain to each entry`
end

Algorithm 2: Pseudocode for sequence generation

- ***assign_longdomains*:** Generates initial assignments for long domains based on the results of the assignability check. These domains are stored in A.

- ***convert_pt*:** Converts the current pairing constraint ($aCFP[i]$) into a pairtable (pt)[4] .
- ***complement(d_{seq})*:** Generates the complementary domain sequence for a given domain sequence d_{seq}. e.g. complement "a b c" →"c* b* a*"

2.4.1 Intuitive understanding of the design scheme

Having demonstrated the effectiveness of the algorithm and design framework, this section provides an intuitive explanation of why the approach is successful.
Each new pairing constraint in the aCFP can induce a multi-step refolding event. This process can be decomposed into an initial refolding step, directly triggered by the newly appended segment, and subsequent refolding steps. This initial step initiates a sequence of zero or more subsequent secondary refolding steps, depending on the folding events that occurred previously. Each new segment added to the sequence introduces, at most, one initial refolding step, while all subsequent steps are inherently accounted for by the design framework. This distinction between initial and subsequent refolding is fundamental to the design approach, demonstrating that one new pair of short domains is necessary to control the refolding process.
Initial refolding is initiated by the newly added segment, causing immediate structural changes. If a pairing constraint requires the structure to refold, then the first refolding rule after extension dictates, where the new pair of short domains is needed in the sequence for the initial refolding. In the algorithm, the initial refolding is initiated during the triggering step. In contrast, **Subsequent refolding** events are triggered by the initial refolding, involving domains that became unpaired during the initial refolding. Due to the consistency of the aCFP, there is only one distinct structure for each combination of paired and unpaired segments. Therefore each scenario has a clear defined structure in accordance to the aCFP.
The algorithm makes use of the two different types of refolding. At the beginning of each step, the algorithm balances competing segments by introducing short domains to create controlled three- or four-way competition, where an undecided structural equilibrium can be shifted toward a target structure, see claim 1. This balancing ensures that no structure dominates prematurely and that the system is prepared for the next transition. Once balanced, a new pair of short domains is added to the targeted segments, initiating the initial refolding. This newly introduced pair acts as a trigger, driving the system toward the desired structural change. All subsequent folding events are inherently predetermined and accommodated by the algorithm's design framework, requiring no additional input. This cycle of balancing and triggering is repeated iteratively for each step in the aCFP until the entire sequence has been processed. By carefully controlling the refolding process in this way, the algorithm ensures that each pairing constraint is satisfied in accordance with the aCFP, maintaining structural consistency throughout. The alogrithm ensures that each subsequence is thermodynamically optimized at each stage, allowing subsequent refolding to follow the previous design steps. In this process, each segment "remembers" the pairing constraints for which it was previously designed.

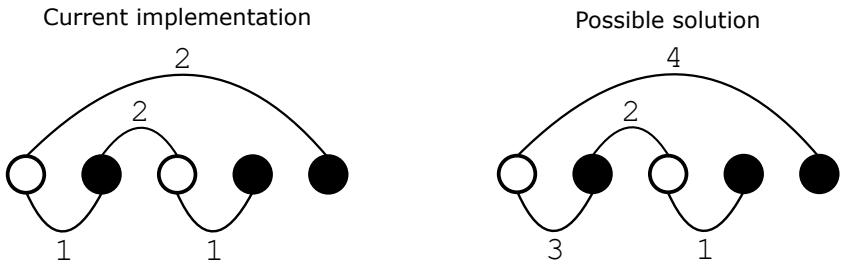


Figure 2.9: Comparison of edge-weight assignments in the current implementation (left) and a possible solution (right). In the current implementation, only the newly introduced edges can receive a higher edge weight, leading to the incorrect selection of structure `.()`. In the proposed solution, a retrospective increase of the edge weight [1,2] allows structure `().` to become favored in C_3.

For example, if a pairing constraint specifies that the first four segments form a single helix, then at all subsequent stages, if these four segments are unpaired, they must continue to form a single helix (see consistency check definition 13). A new structure will only form if another segment is specifically designed to pair with one of these four segments.

2.4.2 sls-translatable paths not covered by the current implementation

As previously mentioned, the current algorithmic implementation of the sls-design scheme does not cover all possible sls-translatable folding paths. A concrete example is the path [`.` → `()` → `().` → `()()` → `(().)`]. Here, the current implementation fails to check consistency due to limitations in the way edge weights are assigned during the consistency check. Since the translation algorithm builds the domain-level sequence based on the same weighting logic, it subsequently fails to produce a valid translation. In the current implementation, only new active edges, which get introduced in the current pairing constraint, can receive a new edge weight. All previously active edges retain their fixed edge weights. This constraint causes the consistency check to fail at step C_3, where the desired structure is `().`, but the structure calculated by the edge weight based nussinov algorithm is `.()` . The reason is that `.()` contains the previously assigned higher edge weight (see figure 2.9). However, it is important to note that there exists an edge-weight assignment for this aCFP-graph that satisfies the consistency requirement. The figure shows a solution in which a retrospective increase in the edge weight of a previous edge would be required to achieve consistency. This suggests that the current implementation is restrictive, allowing only new edges to be assigned a higher edge weight per step, and never modifying previously assigned edge weights. As a result, the implementation fails to verify aCFPs that rely on retroactive edge-weight changes for a successful consistency check and subsequent translation. This example highlights a fundamental limitation: the current implementation of the sls-scheme does not account for all aCFPs that are theoretically sls-translatable. A complete definition of sls-translatability, along with a

generalized strategy for edge-weight assignment and domain-level sequence construction, remains an open challenge.

2.5 Conclusion

This chapter has introduced a novel framework for generating domain-level RNA sequences that satisfy an abstract cotranscriptional folding path (aCFP), allowing a structured approach to sequence design that prioritizes the folding journey rather than just the final structure. Unlike typical RNA design platforms that focus solely on the fully transcribed structure, this framework enables the design of the cotranscriptional folding pathway, providing valuable insight into the kinetics and intermediate states of the folding process. The framework uses three distinct layers for RNA abstraction, starting with a specification layer, where the focus can lie solely on the structural changes of the sequence during transcription. Once a path has been set in the form of an aCFP, analysis on the domain-level can begin, creating a domain-level sequence. Domains are an important intermediary step connecting the specification layer with the nucleotide layer. Using domains allows the design framework to focus solely on the complementarity between different structures and the respective lengths of those structures, without needing to account for specific nucleotide complementarity and base pairing between unwanted structures. Additionally, domain-level analysis allows the breakdown of different types of events that can occur during cotranscriptional folding and reveals how those types of reactions can be induced and controlled. Based on the different types of events that can occur during cotranscriptional folding, a rule set was established, allowing the deconstruction of a cotranscriptional folding path into a sequence of rules and enabling detailed analysis of each individual event during transcription.
The decision to exclude pseudoknots from the framework was deliberate, with the goal of reducing design complexity and simplify validation processes. As discussed in section 1.1.1, the inclusion of pseudoknots increases the computational cost for structural prediction, creating additional challenges for domain and nucleotide level validation. The next chapter introduces a tool for validating domain-level design called CocoSim. CocoSim's dependence on the Peppercornenumerator[30], which does not support pseudoknots, further influenced the decision to omit them in this framework. While the inclusion of pseudoknots would increase the versatility of the framework, particularly in nanotechnologies such as RNA origami[42], the current approach allows for simpler design and validation. Future work could therefore focus on extending this framework to include pseudoknots, potentially opening up broader applications and aligning with the latest developments in nanotechnology.
Theoretically, this design framework can generate domain-level sequences for any sls-translatable aCFP, provided that thermodynamic conditions are met and the sequence has sufficient time to achieve the necessary refolding steps. In practice, however, only a subset of such paths, referred to as coco-translatable, can currently be realized. This subset includes those paths that satisfy three design criteria: assignability, consistency, and favorability. While assignability and traversability are handled robustly, the current imple-

mentation of the consistency check is restrictive. It assigns edge weights incrementally and prohibits reweighting, making it unable to verify paths that require retroactive changes to binding priorities. As shown in chapter 3, this limitation excludes certain theoretically sls-translatable paths, pointing to an open problem in defining a more complete and flexible weight assignment strategy.

Beyond algorithmic limitations, domain-level design becomes increasingly complex as the number of constraints in the aCFP grows. This added length can make subsequent helices more difficult to open and close, complicating the necessary refolding dynamics. To counter this, the framework employs timer domains at the end of each segment, providing a temporal buffer to facilitate refolding before the next transcription step. However, timer domains have practical limits. As they grow longer, they may engage in undesired interactions at the nucleotide level, which can interfere with accurate folding. An alternative approach could be to introduce pause sites, where RNA polymerase halts transcription for a brief period before resuming [43]. This would allow for shorter timer domains, thus enhancing the kinetics and efficiency of the folding process. While this design framework primarily optimizes the design based on thermodynamic properties, kinetics play a crucial role in cotranscriptional folding. To address this gap, a validation tool is essential for confirming that domain-level sequences will follow the desired folding pathway. Since no such validation tool currently exists, this thesis introduces **CocoSim** in the next chapter. CocoSim simulates the cotranscriptional folding path at the domain-level, providing a means to verify domain-level designs before moving forward with nucleotide sequence translation.

3 CocoPaths: A software suite to design and verify abstract folding paths

The **CocoPaths** software package is designed to streamline the process of cotranscriptional folding pathway design and verification by operating across three distinct levels of abstraction (see Fig. 3.1). This multilayered approach bridges the gap between the specification layer and nucleotide sequences, allowing for systematic translation and verification at each step before proceeding with the design.

At the highest level, the abstract cotranscriptional folding pathway (aCFP) represents the desired folding behavior of a nucleic acid sequence in an idealized form. It focuses on the conceptual interactions and structural transitions without specifying the exact sequence or lengths of the individual structural features. This level serves as the blueprint for the intended folding mechanisms and guides the subsequent design process.
The intermediate domain-level acts as the bridge between the abstract level and the nucleotide implementation. At this level, the sequence is represented as a series of domains, distinct subunits with an undefined nucleotide sequence that can independently fold or interact with other complementary domains, each contributing to the overall structural and functional behavior of the sequence. The domain-level allows the verification of the domain-level sequence by using a cotranscriptional folding simulator. With this tool, domain lengths can be adjusted to improve kinetics of the system. Adjustments at this level are essential because correctness has only been established based on the probability of success, not on the time required for refolding. Even if a domain-level sequence aligns with the specifications provided by the aCFP, kinetic simulations may reveal that the desired folding pathway is not feasible in practice due to unfavorable kinetics.
After refining the domain-level design by tuning parameters such as domain length and transcription rate and ensuring kinetic feasibility, the process advances to the lowest level, where a nucleotide sequence is designed based on the domain-level sequence and the information gathered from the simulation. The goal at this stage is to maximize the desired domain-level properties and ensure that the physical nucleic acid sequence will fold as predicted at the domain-level.

The **CocoPaths** software package consists of three software tools for the design process. (i) **CocoPath** generates a domain sequence based from an aCFP, (ii) **CocoSim** simulates cotranscriptional folding of the domain sequence, and (iii) **CocoDesign** designs nucleotide sequences.

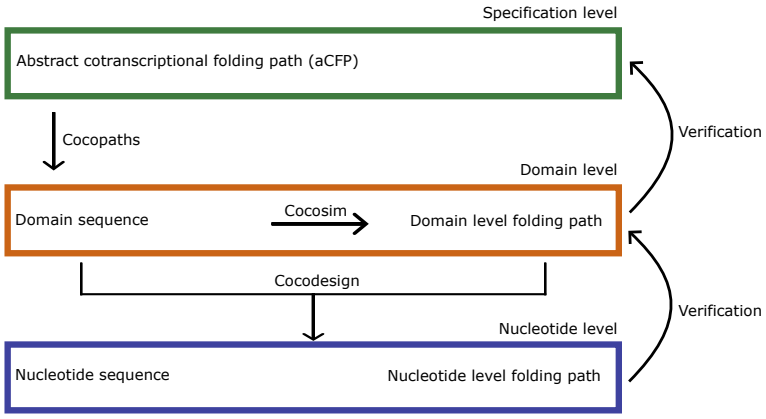


Figure 3.1: Overview of the CocoPaths software package architecture, illustrating the three levels of abstraction: specification level, domain-level, and nucleotide sequence level.

3.1 CocoPath

CocoPath is the first software tool, developed as part of the CocoPaths suite, specifically designed for the construction of domain-level sequences based on an aCFP. It provides a practical implementation of the theoretical framework and algorithms introduced in chapter 2, bridging the gap between abstract folding path design and practical sequence construction. CocoPath fulfills two critical roles in the design process. First it evaluates whether a proposed aCFP is coco-translatable by checking its assignability, consistency and traversability, as defined in section 2.3.1. This evaluation not only determines feasibility, but also extracts complementarities.
The second key role of CocoPath is to use this information to guide the construction of the domain-level sequence. The software leverages the addition of short domains, such as toeholds and clamps, to control cotranscriptional refolding events, ensuring that each step of the folding pathway aligns with the desired structural constraints provided by the aCFP. Toeholds facilitate structural transitions by lowering kinetic barriers, while clamps stabilize intermediate structures to prevent undesired reactions. This approach allows for regulation of the sequence's folding dynamics and guarantees that the thermodynamically optimal structure at each transcriptional step adheres to the aCFP.
While CocoPath focuses on the evaluation of the input and folding path construction, its output is the foundation for all subsequent softwares contained in this package. The domain-level sequences generated by CocoPath are used by downstream tools such as CocoSim to simulate cotranscriptional folding paths that verify the domain-level design, and subsequently by CocoDesign to generate nucleotide-level sequences.

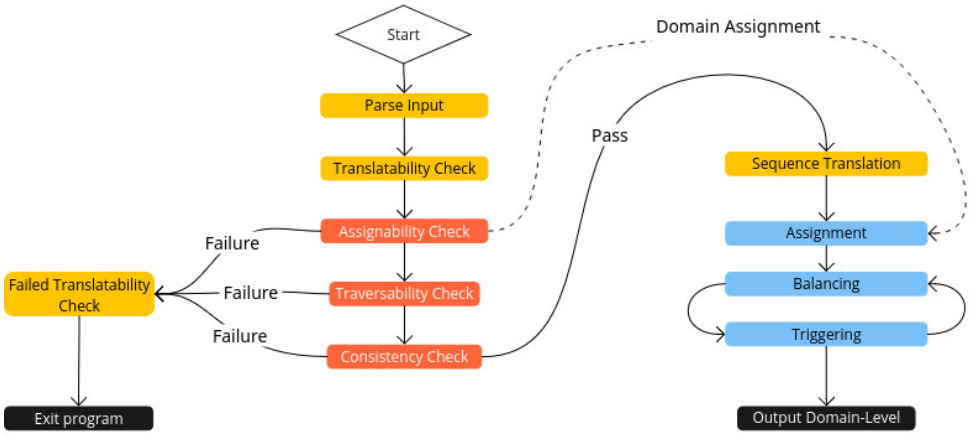


Figure 3.2: A visual representation of the CocoPath workflow. The process begins with parsing the input aCFP, followed by the translatability checks: assignability, traversability and consistency. If the aCFP passes these checks, the domain-level sequence is translated.

3.1.1 Workflow

The workflow of CocoPath is structured to streamline the process of translating aCFPs into domain-level sequences, ensuring that each stage contributes to accurate and efficient sequence construction. The workflow is divided into three main phases: input parsing, translatability check, and sequence generation. Each phase is described in detail below:

Input Parsing

CocoPath begins by parsing the input provided by the user, which typically consists of an abstract cotranscriptional folding path (aCFP). This input defines the desired sequence of structural transitions during cotranscriptional folding.

Translatability Check

The translatability check is a critical step that assesses whether the proposed aCFP can be realized as a domain-level sequence. This phase is divided into three sub checks:

1. **Assignability Check**: Constructs a graph where vertices represent positions in the aCFP and edges represent pairings. The check ensures that the graph is bipartite and verifies that pairings are valid according to the defined constraints. This process also identifies complementary relationships between domains that are essential for the sequence design.

2. **Traversability Check**: Determines whether each transition $C_i \rightarrow C_{i+1}$ in the aCFP can be broken down into a sequence of rules.

3. **Consistency Check**: Ensures that the aCFP satisfies the consistency criterion, as defined in Section 2.3.1.

Domain-level sequence construction

Once an aCFP has been deemed coco-translatable, CocoPath begins constructing the corresponding domain-level sequence. The construction process uses a two-phase approach: a one-time assignment phase followed by an iterative rule application phase. This methodology ensures that the resulting sequence adheres to the folding path defined by the rule sequence, while maintaining both thermodynamic stability and proper folding dynamics at each step.

1. **Assignment:** Creates an array with the initial assignments for long domains based on the results of the assignability check.
2. **Sequence generation:** For each pairing constraint in the aCFP , perform balancing and triggering:
   a) **Balancing**: Ensures that all competing structures at the current step are equally favored by balancing the number of domains.
   b) **Triggering**: Adds domains depending on the refolding event to ensure that the target structure is favored among competing structures.

3.1.2 Future directions for CocoPath

As discussed in chapter 2, the algorithm and design framework underlying CocoPath currently supports only a subset of aCFPs. Specifically, those that are considered coco-translatable. While coco-translatability implies sls-translatability, the reverse is not always true. Thus, not all sls-translatable folding paths can currently be realised by CocoPath. This limitation is mainly due to limitations in the current consistency checking and weight assignment procedures, which cannot handle certain edge weight dependencies required by more complex paths. In addition to this limitation, CocoPath does not support aCFPs containing pseudoknots. The framework operates under the assumption that pseudoknots are absent, in part due to the complexity they introduce in structure prediction and validation. However, it is theoretically possible to extend CocoPath to support pseudoknoted structures. This would significantly broaden its applicability, allowing the design of domain-level sequences for a wider range of RNA folding pathways, including more complex and biologically relevant cases. Addressing these two limitations, extending the coverage of coco-translatability and enabling pseudoknot support, is a crucial step towards increasing the generality and utility of CocoPath in future work.

3.2 CocoSim

Once a domain-level sequence satisfies all the pairing constraints of the abstract cotranscriptional folding path (aCFP), it is essential to verify the design kinetically. While

CocoPath focuses on thermodynamic optimization, it does not account for specific domain lengths or kinetic feasibility. To address this, CocoSim simulates cotranscriptional folding at the domain-level, optimizing domain lengths and refining folding paths based on real time kinetics.
CocoSim is based on a DrTransformer [29] like approach, using all unimolecular reactions and the reaction enumerator from the peppercornenumerator[30]. Each step in the co-transcriptional folding path is simulated using the following steps: (i) expansion of the domain sequence, (ii) enumeration of the system, (iii) adjusting the occupancies in the system, (iv) simulation, and (v) pruning of unoccupied structures.
Through this process, CocoSim generates a detailed cotranscriptional folding path, showing the structural ensemble and calculated occupancies after each domain addition. This allows for kinetic validation and optimization of the folding pathway, ensuring both thermodynamic stability and kinetic feasibility of the design before moving to design of nucleotide level sequences.

3.2.1 Background and Terminology

Cotranscriptional folding can be simulated using various methods, as discussed in section 1.2.2 of the introduction. CocoSim simulates cotranscriptional folding by extending the sequence at each step and calculating the possible structures before continuing. To enumerate the possible reactions, core functions from the Peppercorn software[30] are utilized. Peppercorn, originally designed for DNA strand displacement (DSD) systems at the domain-level, enumerates the reaction space using reaction types such as bi- and unimolecular reactions, facilitating the design process of DSD systems. It takes a set of kernel strings as input and generates all possible reaction products and pathways. Since CocoSim uses core functions from the Peppercorn, similar definitions of various components need to be defined which will be used in CocoSim.

Definition 15. *A **complex** represents an RNA sequence with a specific secondary structure. Each complex has a specific occupancy in the ensmble.*

Definition 16. *A **reaction** $r = (A, B)$ is a tuple consisting of a reactant A and a product B, where A transitions to B. The kinetic rate of the reaction, denoted as k_r, defines the fraction of molecules that undergo the transition per second.*

Reactions are classified based on their kinetic rate k_r into three categories:

- A reaction is *fast* if $k_r > k_{fast}$.
- A reaction is *slow* if $k_{slow} < k_r \leq k_{fast}$.
- A reaction is *negligible* if $k_r \leq k_{slow}$

The threshold values k_{fast} and k_{slow} are tunable parameters that can be adjusted from their default values to refine the classification of reactions.

To represent the reaction space in CocoSim, we use a Single-Stranded Reaction Network (SSRN), where each node corresponds to a secondary structure and edges represent possible structural transitions. Since SSRNs are inherently simpler than Chemical Reaction Networks (CRNs), Peppercorn can efficiently enumerate them.

Definition 17. *A **single stranded reaction network (SSRN)** is a directed graph $G = (V, E)$, where each vertex $v \in V$ represents a single stranded molecule. Each directed edge $e = (v_i, v_j) \in E$ corresponds to a reaction $r = (v_i, v_j)$, where RNA secondary structure v_i transitions into structure v_j with rate $k_r = k_{ij}$.*

The CocoSim algorithm relies on pruning and coarse-graining of domain-level reactions, where negligible and, in some cases, slow reactions are excluded based on the assumption of timescale separation (see Section 3.2.1). In this context, we refer to **macrostates**, which are formally defined below.

Definition 18. *A **macrostate** is a strongly connected component (SCC) within a SSRN. In an SCC, every vertex is reachable from every other vertex in the component, meaning there exists a path between any two vertices within the SCC.*

Macrostates are classified based on the nature of their outgoing reactions:

- *A macrostate is **resting** if all outgoing reactions from the SCC are slow.*
- *A macrostate is **transient** if at least one outgoing reaction from the SCC is fast.*

The strands within a macrostate are classified as resting or transient depending on the classification of the macrostate itself.

Reactiontypes

In accordance with the reaction types previously established in Section 2.1, CocoSim employs these types of reactions to enumerate the system using Peppercorn. Unlike the general Peppercorn implementation, which allows the binding of two separate RNA strands, the enumerator in CocoSim is restricted to unimolecular reactions. This restriction reflects the focus on cotranscriptional folding paths, excluding interactions with external RNA molecules. As a result, all bimolecular reactions were excluded from the implementation in CocoSim.

The enumerator implementation in CocoSim supports the following reaction types:**(I) Bind:** Pairs two unpaired domains to form a new connection. **(II) Open:** Dissociates two previously paired domains, effectively reversing the bind reaction. **(III) Three-way branch migration:** An unpaired domain displaces a paired domain, creating a new pairing. **(IV) Four-way branch migration:** Four paired domains switch their pairing partners to form a new structure. Using these four reaction types, the modified enumerator function from Peppercorn systematically enumerates all possible unimolecular reactions based on the current sequence and structure. Since Peppercorn is based on data derived from DNA, the rates are also DNA based. Although the use of DNA-based rates introduces minor discrepancies, the reaction dynamics of the model are primarily influenced by the variations in domain lengths, which are accounted for by peppercorn.

Timescale separation

CocoSim utilizes the principle of **timescale separation** from Peppercorn to facilitate the enumeration and simulation of the resulting chemical reaction network at each transcription step. In Peppercorn, timescale separation is employed to mitigate the combinatorial explosion in the enumeration of reaction pathways. CocoSim, while also leveraging this method to reduce combinatorial explosion, uses timescale separation to differentiate between fast and slow reactions that may occur between the sequential steps of transcription.

During each transcription step, the RNA sequence grows, and strands formed earlier may undergo reactions before the next domain is added. In this context, strands are assumed to have sufficient time to undergo *fast reactions* (e.g., binding or small conformational changes) before engaging in slower processes (e.g., four-way branch migrations or other more complex structural rearrangements). This separation helps CocoSim accurately model how RNA structures fold cotranscriptionally. Note that the classification of reactions as either fast or slow is determined by the parameters k_{slow} and k_{fast}, which are based on default values that can be adjusted depending on the system being simulated. This flexibility ensures that CocoSim can be modified to suit the specific system being simulated.

Reaction enumeration

At the start of each transcription step, every complex in the current chemical reaction network is extended by the next domain to be transcribed. These extended strands serve as the new input for the enumerator, which enumerates all feasible reactions based on the current secondary structures. Although the domain-level sequence generated by CocoPaths is designed to follow controlled cotranscriptional folding paths, where a single complex should dominate the structural ensemble at specific transcriptional steps, there can be folding paths where multiple strands coexist between these steps. This is acceptable as long as the target structure of the abstract cotranscriptional folding path is dominant after transcription of the timer domain.

3.2.2 Algorithm

CocoSim's simulation process follows a cyclic algorithm designed to model cotranscriptional folding at the domain-level. Each cycle represents the addition of a domain to the sequence and the subsequent reactions that can occur as a result of this extension. The algorithm iterates through a series of steps, progressively refining the system until the entire domain-level sequence has been processed. A visual representation of the algorithm can be seen in figure 3.3

The five key steps of the CocoSim algorithm are as follows:

I. Extension of the domain sequence

II. Enumeration of the System

III. Updating occupancies in the system

IV. Simulating kinetics of the system

V. Pruning of unoccupied structures

I: Extension

The **extension step** marks the beginning of a new cycle in the cotranscriptional folding simulation. This step represents the addition of a new nucleotide during transcription or, in the case of CocoSim, the addition of an entire domain, which corresponds to multiple nucleotides being added at once. During the expansion step, the next domain in the sequence is appended to each complex in the current ensemble. The transcription speed is dependent on the length of the appended domain. At this stage, the each complex in the system is printed to the terminal before proceeding to the enumeration phase.

II: Enumeration of the system

Once the system has been extended, exploration of the structural space begins. The addition of the new domain allows for the formation of new secondary structures. To explore this space, the *enumerate* function from the Peppercorn is utilized. This function takes the current system, represented by the extended strands, as input. The system is enumerated, and possible reactions between the new strands are calculated.
Depending on the reaction rates and the predefined kinetic parameters k_{slow} and k_{fast}, strands are classified as either *transient* or *resting*, according to the macrostate they belong to (see Definition 18). After the enumeration step, the system now consists of updated strands with new or modified secondary structures, along with newly calculated reaction rates between these structures.

III: Updating occupancies of the system

After the enumeration step, the occupancies of the system need to be updated. According to Definition 18, transient strands are characterized by fast outgoing reactions. CocoSim assumes that fast reactions occur instantaneously. As a result, a redistribution of the occupancies of transient strands is necessary. This redistribution accounts for all fast reactions, ensuring that the system is adjusted such that only resting states retain non-zero occupancy. Transient states are effectively excluded from the system, as their occupancies are redistributed and reduced to zero.
This redistribution occurs in two stages. First, the occupancy of each transient complex is redistributed among the connected resting macrostates according to the decay probabilities provided by Peppercorn. These probabilities determine how much of each transient state's occupancy transfers to the neighboring resting macrostates. In the second stage, the strands within each resting macrostate are updated. This is done by using the stationary distribution of the macrostate, ensuring that the total macrostate occupancy is appropriately allocated across its strands.

By following this process, the system retains only resting strands with non-zero occupancies, reflecting the outcomes of fast reactions, while transient states are fully accounted for and effectively removed from the system. This results in a system where fast reactions occur exclusively within a resting macrostate and macrostates are connected only via slow reactions. These slow reactions will be addressed and accounted for in the next step of the simulation process.

IV: Simulation of slow reactions

During the time between adding new domains, slow reactions between macrostates can take place. To simulate these reactions, CocoSim uses a modified version of the *Pilsimulator* script from the Peppercorn package, which solves ordinary differential equations (ODEs) using SciPy's ODE solver [44]. The system of ODEs is derived from the condensed reaction network, where each reaction represents the interactions between macrostates and is characterized by specific reaction rates and initial concentrations. The duration of the simulation step is dependent on the length of the recently added domain. Following the simulation, the occupancies of all strands are adjusted based on the values calculated by the ODE solver, ensuring that the system accurately reflects the new state of each complex.

V: Pruning of unoccupied structures

To mitigate combinatorial explosion and avoid enumerating unoccupied secondary structures, CocoSim implements a pruning step at the end of each cycle. By default, the cutoff threshold is set at 5%, meaning the least occupied structures are progressively pruned until the total occupancy removed reaches this threshold. However, no single secondary structure is pruned if its removal would exceed the cutoff limit, ensuring that significant structures are preserved. Pruning is performed at the macrostate level, as pruning individual strands within a macrostate would be counterproductive. Since reactions within a macrostate are assumed to occur instantaneously, pruning individual strands that fall below the threshold is unreasonable. As a result, only entire macrostates are considered for pruning.
When a macrostate is pruned, its occupancy must be redistributed. This process occurs only if the pruned macrostate has outgoing condensed reactions. If the macrostate contains outgoing condensed reactions, its occupancy is redistributed proportionally among the connected macrostates, according to the outgoing kinetic rates. If there are no outgoing condensed reactions, the macrostate is not removed. This approach of pruning structures only when outgoing condensed reactions are present could potentially cause problems in certain cases. In practice, however, the likelihood of encountering a to be pruned state with no outgoing reactions is extremely small. This condition is included as a precautionary measure to handle such rare scenarios if they do occur. CocoSim utilizes the *condense* function from Peppercorn to help with this process. By manually adjusting the reaction rates of the outgoing reactions from the macrostate to a value higher than k_{fast}, Peppercorn classifies the pruned macrostates as transient. It then recalculates the

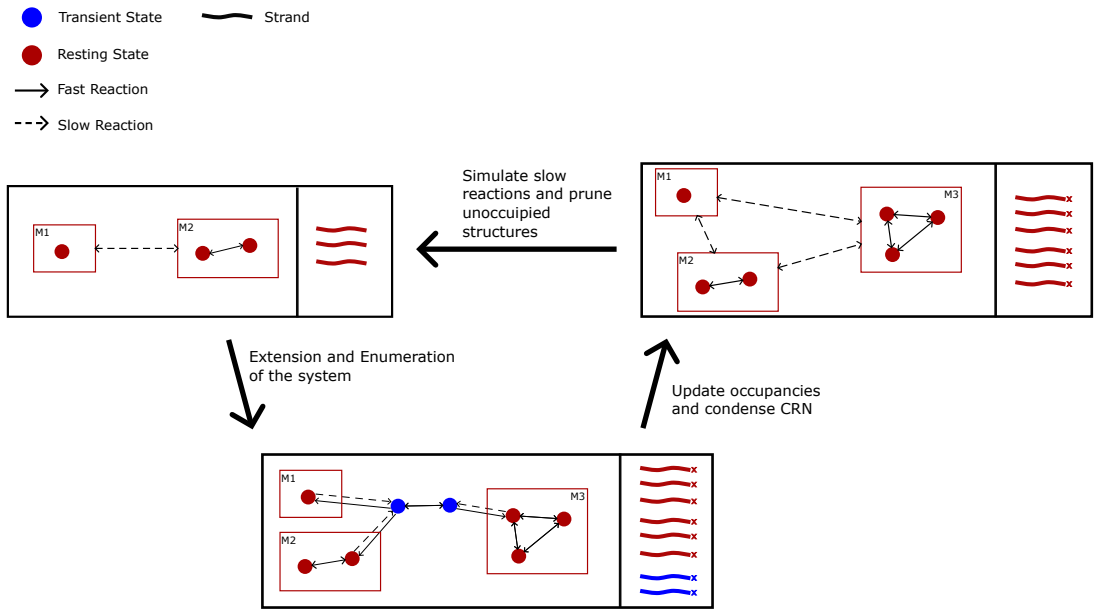


Figure 3.3: A graphical visualization of the CocoSim algorithm. The process begins with the extension step, where a new domain (indicated by X) is added to the sequence, followed by enumeration of the system to explore the structural space and create a chemical reaction network. Transient states (blue) and resting states (red) are connected via fast reactions (solid lines) and slow reactions (dotted lines). The algorithm updates the occupancies of states, condenses the chemical reaction network and simulates slow reactions between macrostates. Finally, pruning eliminates unoccupied or low-occupancy macrostates while preserving computational efficiency. Each cycle iteratively follows these stages.

occupancies of the entire SSRN without the pruned macrostates.

3.2.3 Output format

CocoSim displays the cotranscriptional folding results after each extension step by showing the secondary structures along with their corresponding occupancies. At this point, the newly added domain is unpaired, showing the structures before the simulation has taken place. Whether CocoSim is used with a sequence designed by CocoPath or not, an additional logic structure view can be provided. This highlighted view focuses solely on the binding interactions of the long domains, offering a clearer understanding of whether the sequence design has been successful at the domain-level. This feature aids in quickly assessing the correctness of the folding path and domain interactions.

3.2.4 Optimizing kinetic properties of the domain-level sequence

Apart from simulating and verifying the domain-level folding path, one of CocoSim's core functionality lies in optimizing the kinetic properties of the domain-level sequence, such as domain lengths and transcription rates. By adjusting the length of individual domains, the user can significantly influence the folding dynamics of the sequence. One of the simplest adjustments is to change the length of the timer domains in each segment. The primary purpose of a timer domain is to provide sufficient time for the upstream sequence to reach its thermodynamic optimum. As such, the length of each timer domain must be tailored to match the time required for the upstream sequence to fold into its target structure. Since this timing is not universal, each timer domain must be individually tuned.
The default formula for the length of a timer domain is based on its position within the sequence, where the length of each timer is calculated as the position of the timer starting with 0 (e.g., the first timer is at position 0, the second is at position 1, and so on) multiplied by 4. This formula assumes that, on average, each timer domain requires an increase in length of 4 to facilitate the refolding process. However, it is recommended to customize the lengths of individual timer domains, as certain transcription steps may require longer or shorter timer to effectively optimize folding dynamics.
The adjustment of timer lengths is not constrained by any strict limits in terms of length. However, this flexibility is not applicable when modifying short or long domains. CocoPaths assigns specific lengths to short and long domains based on their targeted kinetic properties. Long domains are typically longer, which makes them less likely to unbind spontaneously. In contrast, short domains are designed to bind and unbind more readily due to their shorter length. Consequently, any adjustments to the lengths of these domain types should be made with caution to avoid disrupting their intended roles within the sequence and affecting the overall folding behavior.
Transcription speed is another critical factor influencing the cotranscriptional folding path. During transcription, the RNA sequence progressively elongates, and the rate at which this occurs depends on various conditions. As discussed in Section 1.1, transcription speeds can range from 10 to 200 nucleotides per second [21]. This variability significantly impacts the folding process, as faster transcription speeds allow less time for structural rearrangements between steps. Consequently, longer timer domains are required to accommodate faster rates, ensuring that refolding events have sufficient time to occur.
When using CocoSim, it is essential to consider transcription speed to adjust domain lengths that align with the experimental conditions. Selecting transcription parameters that reflect the anticipated speed during experimental validation ensures that the sequence is optimally tailored to achieve the desired folding path.

3.2.5 Limitations and future work

CocoSim serves as a domain-level cotranscriptional folding simulation tool, enabling detailed analysis and optimization of domain-level RNA folding processes. Using Peppercorn, CocoSim provides a robust framework for reaction enumeration and kinetic simulation,

allowing users to refine domain-level sequences with a focus on thermodynamic stability and kinetic feasibility. However, despite its strengths, CocoSim has certain limitations that need to be addressed to expand its applicability and effectiveness.
Another limitation is the exclusion of pseudoknots from the design framework. As discussed in previous sections, pseudoknots are crucial structural elements in many RNA designs and are widely used in nanotechnology applications [42]. Currently, CocoSim relies on the enumerator function from Peppercorn, which was originally designed for DNA strand displacement systems and does not support pseudoknot formation. While this exclusion simplifies reaction enumeration and simulation, it significantly limits the range of structural features that can be modeled. Inclusion of pseudoknot support would require updates to the Peppercorn to allow it to account for these complex interactions. This enhancement would allow CocoSim to simulate and validate a broader class of RNA folding paths, making it more versatile for advanced RNA design challenges.
Another limitation of CocoSim is its reliance on kinetic rates derived from DNA reactions, which may not fully capture the nuances of RNA folding dynamics. Although this is not a significant problem in scenarios where domain lengths primarily govern folding kinetics, it can lead to inaccuracy in the case of complex RNA structures or highly specific kinetic behavior. Future iterations of CocoSim could address this issue by basing reaction rates on experimental data derived from RNA experiments.

3.3 CocoDesign

CocoDesign generates a nucleotide sequence from the domain-level sequence while maintaining its intended folding path. At the start of CocoDesign, the domain-level folding path is extended into a nucleotide level folding path, which incorporates the specific characteristics of each domain. This nucleotide path is referred to as the **extended folding path**, where each structure represents a string representation of a pairing constraint. The nucleotide level allows both theoretical[29, 24] and experimental validation[45, 46]. Using the domain-level sequence and the folding path simulated by CocoSim, CocoDesign generates a nucleotide sequence that satisfies the constraints set at the domain-level and follows a cotranscriptional folding path that is consistent with the domain-level folding path used as input. The sequence is designed iteratively, with each step sampling and evaluating a new nucleotide sequence based on different parameters and ultimately, the best scoring sequence is returned.

3.3.1 Background and Terminology

Infrared package

Infrared[47] is a framework designed to address a wide range of sampling and optimization problems, especially in bioinformatics. It is built around the concept of feature networks, which consist of variables, constraints and functions that define the sampling space from which nucleotide sequences can be sampled. This concept makes Infrared particularly

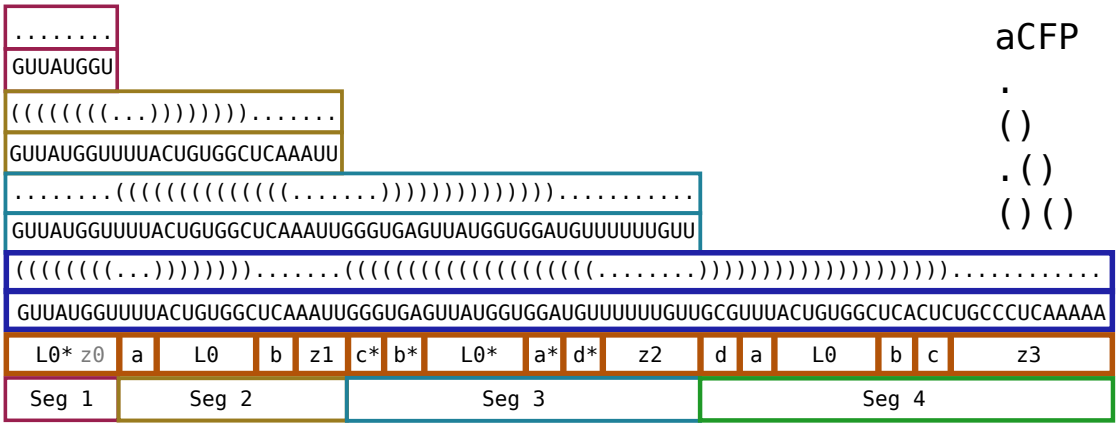


Figure 3.4: Visualizes the relationship between the domain-level and the nucleotide level during design. The bottom three levels represent the designed nucleotide sequence, the domain-level sequence and the corresponding segments. Above the nucleotide sequences are always the corresponding secondary structures.

suitable for problems with complex dependencies, such as RNA sequence design with multiple target structures and dependencies.
Through its Python interface, CocoDesign uses Infrared to create a sampling space for generating nucleotide sequences. Infrared's sampler plays a critical role in this process, efficiently generating nucleotide sequences that satisfy the defined constraints imposed by the domain-level. In the context of CocoDesign and the Infrared package, the sampling space is the collection of all nucleotide sequences that satisfy the constraints imposed by the domain-level. Further specification of how the constraints are defined is given below.

Constraints

For efficient nucleotide sequence sampling, infrared allows the application of constraints on the sampling space to reduce its size. CocoDesign applies two constraints on the sampling space. The first is the **identical domains constraint**, which ensures that each sampled sequence obeys the constraints imposed by the domain-level and that domains remain true to their definition, see def 5. Let D represent the set of domain names. For each domain $d \in D$, let $L(d)$ denote the length of that domain. The nucleotide sequence is denoted as S. Additionally, let $K(d)$ indicate the starting indices in the nucleotide sequence corresponding to the domain d.

The identical domains constraint states that for any two instances of the same domain d, denoted as d_i and d_j (where $d_i = d_j$), the following condition must hold:

$$S[K(d_i) + m] = S[K(d_j) + m] \quad \text{for all } m \in [0, L(d) - 1].$$

This condition implies that for every nucleotide position m within the domain, the nucleotide in the sequence S at the index $K(d_i) + m$ must equal the nucleotide at the

index $K(d_j) + m$. This relationship must be satisfied for all pairs of indices i and j corresponding to the same domain d.
The second constraint is the **folding path constraint**, which ensures that all intended secondary structures are feasible in the nucleotide sequence. Let P represent the sequence of secondary structures in the extended folding path, where each structure $p \in P$ specifies the base pairs B_p in that structure. For each base pair $(i, j) \in B_p$, the constraint guarantees that the nucleotides at positions i and j in the final nucleotide sequence must form one of the following valid base pairs: “AU", “CG", “GC", “GU", “UA", or “UG". This ensures that the nucleotide sequence supports the intended secondary structure by maintaining correct base pairings at the specified positions.

In accordance with the constraints provided in the ViennaRNA package [4], CocoDesign differentiates between two different pairing constraints:

- Relaxed constraint("."): In this scenario, unpaired nucleotides are free to adopt any secondary structure as long as they do not interfere with the specified pairings. This flexibility allows for a broader range of successful designs, as minor deviations in unpaired regions do not impact the overall folding pathway. Timer domains, in particular, benefit from this constraint, as enforcing strict unpairing over long sequences is inherently challenging. By permitting structural variability in unpaired regions, the relaxed constraint enables more feasible and adaptable designs.
- Fixed constraint ("x"): Here, unpaired regions must strictly remain unpaired throughout transcription. This imposes a more rigid structural requirement, making it more challenging for sequences to meet the success criteria.

By offering these two levels of constraint strictness, CocoDesign provides precise control over the sequence design process. The relaxed constraint allows greater flexibility, making it easier to generate viable sequences, while the strict constraint ensures stricter adherence to the intended folding path.

3.3.2 Optimization process for nucleotide sequences in CocoDesign

The creation of CocoDesign centered around three foundational components. First, constraints had to be defined to properly set up the Infrared model, ensuring it could sample nucleotide sequences that fit the specified requirements. With the sampling space in place, the next step was the design of an objective function capable of evaluating candidate sequences. This function must effectively assess sequence quality based on the structural and kinetic properties necessary for successful folding. Finally, a method to explore the solution space and iteratively refine sequences must be implemented, ensuring an efficient search for optimal designs.

In the optimization process, finding optimized nucleotide sequences requires a reliable objective function to evaluate sequences efficiently. A key challenge is finding the right balance between speed and accuracy, particularly for sequences designed by CocoDesign

where kinetics significantly influence fitness. The objective function must assess how well a sequence satisfies the extended folding pathway on which it is based. Although running cotranscriptional folding simulations would directly verify whether a sequence satisfies the structural constraints of the extended folding path, this method is computationally expensive and impractical for evaluating a large number of sequences. As a result, the objective function for CocoDesign needs to approximate the cotranscriptional folding process with minimal computational cost. A combination of thermodynamic and kinetic parameters is used to capture essential features of the folding dynamics.
The goal of the design is to obtain a sequence that mimics the domain-level folding path at the nucleotide level, based on the structural constraints as specified in the extended folding path. The objective function must take into account each of these constraints to verify that each individual constraint has been met. At each step of the extended folding path, a subsequence is taken from the full-length nucleotide sequence, with a length equal to the length of the constraint for that step.

Objective function

After extensive iterations with different parameters, the final objective function for evaluating each step is based on two key aspects: a thermodynamic term and a kinetic term. Balancing these two components ensures a comprehensive and accurate evaluation. The thermodynamic parameter calculates the difference between the ensemble free energy of the sequence and the constrained ensemble free energy of the sequence, where the ensemble is constraint to structures fulfilling the structural constraints of the extended folding path. Minimizing this term ensures that the thermodynamic ensemble is dominated by structures that align with the structural constraints of the extended folding path, thereby increasing the probability that the sequence will adopt the desired structure.
The kinetic term of the objective function focuses on the energy barrier required to transition between consecutive target structures. Using the *path_findpath* function from the ViennaRNA package [4], it calculates the energy barrier for folding from one structure to the next. Due to the difference in length between the two target structures, unpaired nucleotides are appended to the shorter structure until both are of equal length. Minimizing this term helps reduce the energy barrier between structures, enabling faster refolding events.

Let C_i represent the constraint at step i in the extended folding path, and let χ denote the nucleotide sequence. For each constraint C_i, there exists a corresponding subsequence χ_i of χ, where χ_i satisfies the structural constraint imposed by C_i.

The objective function O_i for each subsequence χ_i is formulated as:

$$O_i = G(\chi_i) - G_c(\chi_i, C_i) + 0.1 \cdot b(\chi_i, C_{i-1}, C_i) \tag{3.1}$$

Where:

- $G(\chi_i)$ is the ensemble free energy of the subsequence χ_i at step i,

- $G_c(\chi_i, C_i)$ is the constraint ensemble free energy of the subsequence χ_i with the constraint C_i,
- $b(\chi_i, C_{i-1}, C_i)$ is the free energy barrier for the sequence at step i going from constraint C_{i-1} to C_i.

For the whole sequence, the objective function is given by minimizing the maximum value across all calculated steps, ensuring that computational efforts focus on the current worst part of the designed path:

$$O = \min\left(\max_{i=1}^{C}\left(G(\chi_i) - G_c(\chi_i, C_i) + 0.1 \cdot b(\chi_i, C_{i-1}, C_i)\right)\right) \tag{3.2}$$

The objective function integrates both thermodynamic and kinetic parameters to evaluate a nucleotide sequence. The factor of 0.1 was chosen because at later stages in the optimization the energy barrier typically ends up in ranges from 3 to 10 kcal/mol, whereas the difference in ensemble energy often falls between 0 and 1 kcal/mol. To ensure both terms are given equal weight at the end of optimization, a factor of 0.1 is applied to the kinetic term, balancing the contributions of both parameters. Additionally, the objective function takes the maximum value out of all calculated steps to prevent cases where a few values are highly optimized while others are neglected.

Sampling sequences using Monte Carlo simulations

To generate nucleotide sequences from the sampling space, CocoDesign employs a Monte Carlo optimization strategy to explore potential solutions. This method introduces an element of randomness into the optimization process, facilitating broad exploration of the solution space. By iteratively sampling candidate sequences, evaluating them with an objective function and probabilistically accepting or rejecting changes, this approach avoids becoming trapped in local minima. This randomness allows for a more comprehensive search, increasing the likelihood of finding optimal solutions that satisfy the defined constraints.
In CocoDesign, the resampling function from Infrared [47] is used to generate solutions relative to the current best, focusing the optimization on specific regions of the solution space. This targeted approach uses connected components of the model to localize resampling, ensuring that changes remain near the currently optimal solution. Sequences that perform worse than the current best are still accepted with a probability determined by the score difference and a temperature parameter. This probabilistic acceptance criterion enables the algorithm to escape local optima and maintain a balance between exploration and refinement during the optimization process.

Algorithm

This section describes the algorithm developed to translate a domain-level sequence and domain-level folding path into a nucleotide sequence using infrared. The algorithm begins by generating a nucleotide level folding path corresponding to the extended folding

path based on the domain-level folding path. It then initializes the infrared model and refines the sampling space by adding constraints for identical domains and folding path to ensure that only valid sequences are sampled. Once the sampling space is defined, the optimization process begins by sampling a random sequence and evaluating it using the objective function (see Equation 3.1). The sequence is accepted as the new best sequence either if it improves the score or with a probability based on the score difference and the temperature. This process is repeated for n iterations, resulting in the return of the best scoring nucleotide sequence along with its score.

Input: *d_seq*: domain-level sequence, *d_path*: domain-level folding path, *d_lengths*: domain lengths (optional), n: number of sampling steps, *temp*: temperature
Output: Best scoring nucleotide sequence and its score
Step 1: Generate nucleotide level folding path
Extend the domain-level folding path into a nucleotide-level folding path ($extended\_folding\_path$) based on the domain lengths;
Step 2: Model creation
Initialize a model using Infrared based on the nucleotide sequence length derived from the domain lengths and domain sequence;
Apply constraints to the model:

- Identical domains constraint
- Folding path constraint

Step 3: Optimization
Sample a new nucleotide sequence;
for $j \leftarrow 1$ **to** n **do**
  Calculate the objective function score($O(seq)$) for the sampled sequence;
  if $O(seq_{new}) < O(seq_{old})$ *(i.e., lower than the current best)* **then**
    Accept the new sequence;
  else
    Accept the sequence with a probability:

$$p = \exp\left(\frac{O(seq_{new}) - O(seq_{old})}{temp}\right)$$

    ;
  end
  Sample a new nucleotide sequence using the *resampler* function from Infrared[47];
end
Step 4: Output the best scoring sequence
Return: best scoring sequence;

Verifying Nucleotide designs

After a nucleotide sequence has been designed, the question lies in how successful has the design been. For this various existing cotranscriptional folding simulators like DrTransformer [48] or Kinfold [24] can be used to simulate a cotranscriptional folding path. Determining the success of the nucleotide design can be done by comparing the simulated structures at each step corresponding to the step in the extended folding path. By looking at the occupancy of the targeted structure at the specific sequence length, an assertion wether or not the design was successful can be made. In this thesis, a nucleotide level design is defined as successful if it satisfies the pairing constraints specified by aCFP. CocoDesign distinguishes between two levels of pairing constraints: a relaxed constraint ("."), where unpaired nucleotides may adopt any secondary structure provided they do not interfere with the specified pairings (which benefits timer domains), and a fixed constraint ("x"), where unpaired regions must remain strictly unpaired throughout transcription. See section 3.3.1 for more information.

3.3.3 The objective function journey

In the search for a better fitting objective function, various parameters were tested in different configurations to find one whose value correlated well with the fitness of the simulated nucleotide results. Nevertheless, the current objective function is the best fitting one resulting from a series of attempts to find the best fitting ones. The following parameters were also considered and included in the objective function during testing but excluded in the final version:
The **Ensemble defect**[32] was included in the objective function during testing to further improve the thermodynamic optimization of the sequence to exhibit specific structures during specific transcription lengths. The intended effect of minimizing the ensemble defect was to potentially exclude unfavorable structures at each step. However, the ensemble defect was ultimately excluded from the objective function due to its negligible contribution to sequence fitness. The difference in ensemble free energy already accounted for most of the factors that the ensemble defect would address. Including the ensemble defect added complexity without providing significant additional benefits. To maintain simplicity and computational efficiency in the objective function, it was deemed unnecessary to include this term.
The **mean square error (MSE)** was an additional parameter tested within the objective function. Initially, the score was calculated as the sum of each subscore, but this sometimes resulted in optimizing only a few steps while leaving others almost completely neglected, leading to folding paths where the optimized steps dominated the folding path while the ensemble at the neglected steps didn't satisfy the structural constraint. To address this problem, MSE was introduced to penalize significant variations in the subscore values. By incorporating the MSE, the subscores became more consistent and large deviations were minimized. However, an unintended consequence was that any step with a naturally better score could be penalized due to the averaging effect of the MSE. As a result, the objective function was eventually set to minimize the maximum subscore. This approach

serves a similar purpose to the MSE by striving for low scores across the board while allowing some variation between steps.

An important thing to note is that the objective function as stated in equation 3.2 correlates only to a certain degree with the outcome of cotranscriptional folding simulators on the nucleotide level. Meaning that a lower score of the objective function does not always result in a sequence with a better cotranscriptional folding path. Where a better cotranscriptional folding path means that the intended structures at the specific lengths during transcription observe a higher occupancy.
A possible reason for the discrepancy between the objective score and the fitness of the sequence could be that the optimization process only optimizes certain sequence lengths of the folding path and the transitions between these steps. On the thermodynamic side, the objective function optimizes only the structures at each step. The transition between these structures is not included and is only approximated by calculating the kinetic barrier between these two structures. By optimizing and thus reducing the energetic barrier between structures, the objective function strives for a fast and energetically easy transition from one structure to another. An analysis of well scored nucleotide sequences where the folding pathway deviated from the intended design revealed a problem in the folding kinetics. In some cases, the optimization process produced an intermediate structure where the energetic barrier required to transition into the target structure is kinetically unfavorable. Figure 3.5 illustrates this problem. Although the overall kinetic barrier is reduced compared to previous iterations, the folding path is trapped in a local energy minimum. The energy barrier required to transition from this local minimum to the target structure is too high, preventing the transition from occurring in the required time frame. This results in the system being trapped in an unwanted state where further progression along the folding pathway is kinetically unfavorable. Addressing this issue in the objective function would significantly increase the computational workload required to evaluate each sampled nucleotide sequence. Incorporating intermediate steps into the objective function could result in a more precise and realistic representation of the folding process, better reflecting real-world dynamics. However, this approach would drastically raise computational time, as additional calculations would be needed for each sample to account for more steps. Balancing this potential improvement in accuracy with the associated increase in computational time remains a key challenge.

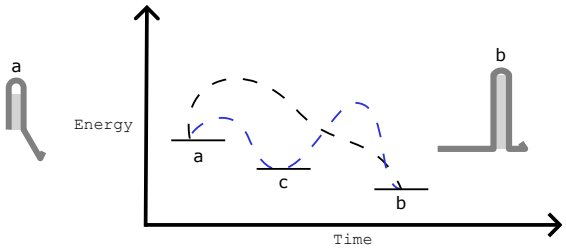


Figure 3.5: Visualization of the folding path issue during optimization. The starting structure (a) transitions toward the target structure (b). The black dashed line represents the previous path, while the blue line shows the current pathway. In the current path, the intermediate structure (c) forms a local energy minimum, introducing a high energy barrier between the intermediate structure and the target structure. This barrier prevents efficient transition to the target structure within the required time frame.

4 Testing CocoPaths

The design framework and software package developed in this thesis provide a comprehensive foundation for cotranscriptional folding path design. This framework employs three layers of abstraction to systematically guide the design process. At the first layer, the specification layer, abstract cotranscriptional folding paths are defined as sequences of pairing constraints. This layer emphasizes on the structural rearrangements occurring during transcription, disregarding structural lengths to simplify the representation.
The second layer, the domain level, introduces structural length, enabling the construction of domain level sequences that adhere to the specified rearrangements during transcription. This level also allows for domain level simulations of cotranscriptional folding paths, providing a means to evaluate the feasibility and accuracy of the design.
Once a domain sequence passes evaluation, it can be translated into a nucleotide sequence at the nucleotide level, the third layer of abstraction. At this stage, the design can be analyzed using theoretical tools for computational validation and experimentally for practical testing.
The tools developed in this thesis, **CocoPath**, **CocoSim** and **CocoDesign**, implement these principles and facilitate the design, simulation, and optimization of sequences across all abstraction layers. Together, these tools create a robust framework for advancing cotranscriptional folding path design from theoretical concepts to practical application.
While the framework is conceptually robust at each abstraction level, theoretical validation alone is not sufficient. We have shown symbolically, without considering domain lengths, that the design algorithm works correctly. To fully evaluate the potential of the CocoPaths system, it must be tested at all levels of abstraction, validating the interplay between the specific layers. This chapter shifts the focus to applied testing, evaluating the performance of the framework at different levels of abstraction using a wide range of folding paths. Through statistical analysis, we examine the functionality of the CocoPaths package across all layers, ensuring that the theoretical design holds in practical implementation.

4.1 Space analysis of designable paths

Before analyzing individual paths based on the three checks and finally evaluating their coco-translatability, an exploration of the combinatorial space was performed. For this purpose, a script systematically generated all valid pairing constraints for each sequence length. The complete set of aCFPs was then constructed by computing the cartesian product of these pairing constraints over all sequence lengths. This set of valid aCFPs was then subjected to the three checks described in section 2.3.1. The verification process included in the CocoPaths software was used to evaluate all three criteria. The results

for aCFPs up to length 7 are summarized in Table 4.1, which illustrates the different set sizes for the three criteria.
While the total number of valid aCFPs (Σ) increases exponentially with sequence length, the percentage of paths that satisfy all three constraints (A ∩ C ∩ T) decreases sharply. Assignability serves as the primary filtering criterion for coco-translatability. However, the additional constraints of consistency and traversability further narrows down the set of viable paths, ultimately permitting only a small fraction of all initially valid aCFPs. This trend highlights the increasing difficulty of generating fully coco-translatable folding paths as sequence length grows. Due to the definitions of consistency def. 13 and traversability def. 14, which requires assignability, the sets containing C only, T only and $C \cap T$ are empty.

Length	Σ	∅	A only	A ∩ T	A ∩ C	A ∩ C ∩ T
3	8	0	2	0	0	6
4	72	3	45	2	0	22
5	1 512	214	1153	51	2	92
6	77 112	23 814	52 082	756	42	418
7	9 793 224	5 038 338	4 739 179	13 171	498	2 038

Table 4.1: Combinatorial counts of valid aCFPs up to length 7 (only nonzero conditions shown). In this table, the rows represent the sequence lengths and the columns represent the outcomes of the combinatorial checks: assignability (A), consistency (C), and traversability (T). The column Σ denotes the total number of paths, and the column labeled A ∩ C ∩ T shows the number of coco-translatable paths (i.e. paths that satisfy all three checks) at that length. The omitted columns ("C only", "T only", and "C ∩ T") were excluded because they are zero for all lengths.

4.2 Statistics over all designable folding paths of the length six

A further analysis of all coco-translatable folding paths was performed using the complete CocoPaths software suite. This testing aimed to evaluate the usability and performance of the whole software package, while examining their interactions within the framework. The analysis provides insight into the strengths and limitations of the CocoPaths package and highlighted areas for refinement. Evaluation of the resulting sequences with DrTransformer[29] served as a benchmark for assessing the overall success of the design process. Key metrics, such as the feasibility of designs at both the domain and nucleotide level, provided valuable feedback for improving the design framework and its practical applicability.
The general workflow of the test starts at the specification level, where an aCFP gets taken from the set of coco-translatable aCFPs and then a domain level sequence for

that aCFP gets created. Afterwards the domain level sequences is simulated using the cotranscriptional folding simulator from CocoSim. Following this, a nucleotide level design is generated using CocoDesign, resulting in a nucleotide sequence. Once the optimization process in CocoDesign is complete, and the best-scoring sequence is returned, the cotranscriptional folding path of this sequence is simulated using DrTransformer [29]. The DrTransformer output is then analyzed to determine whether the simulated folding path satisfies the pairing constraints as indicated by the aCFP from which the sequence was based on.
A sequence design is considered successful at both levels if the target structure of each pairing constraint achieves an occupancy of more than 50% at specific transcript lengths during transcription. This ensures that the desired folding pathway is followed with sufficient stability. As mentioned in section 3.3, CocoDesign uses 2 different levels of strictness applying the folding path constraint. Therefore, both constraints, relaxed and fixed, are applied and analyzed separately.
Since the design process is heuristic, the workflow allows multiple attempts to achieve a successful design. The system allows up to 20 attempts to generate a valid nucleotide sequence. To conserve computational resources, the process exits once a successful nucleotide sequence design is achieved. The domain lengths of the sequence were kept at their default values and remained unchanged across different aCFPs, meaning they were not individually optimized for each aCFP.

The dataset used for the analysis was created by using all coco-translatable aCFPs of length six from section 4.1, resulting in 418 coco-translatable aCFPs. This length was chosen to balance the comprehensiveness of the dataset with the computational demands of the design and simulation process. Extending the analysis to aCFPs of length seven would lead to a combinatorial explosion, with the number of designable paths increasing to 2 038. The primary time limiting factor in the process is the nucleotide level design, where both the translation process and the cotranscriptional folding simulation require significant computational resources. Given that multiple attempts may be needed to achieve a successful design at the nucleotide level, the computational cost of analyzing longer folding paths becomes impractical. For this reason, only folding paths up to a length of six were included in the extensive study. Below is the result of the analysis for the set of designable aCFPs of length six.

Results

The data presented in Table 4.2 was acquired using both variants of the folding path constraint (relaxed and fixed). It shows that, with the exception of one case, all domain-level designs are successful. At the nucleotide level, the relaxed constraint resulted in successful translation for 76% of the aCFPs, while the fixed constraint resulted in successful translation for 65% of the cases. Table 4.2 also provides a detailed breakdown of success counts at both the domain and nucleotide level for different numbers of refolds. The number of refolds indicates how many initial refolds(see sec. 2.4.1) occur in the aCFP and directly affects the complexity of the design. Domain-level success rates generally remain

high and stable across different numbers of refolds. However, nucleotide-level success decreases significantly with increasing refolds under both relaxed and fixed constraints. Under the relaxed constraint, nucleotide success rates decrease from 98% (zero refolds) to 31% (four refolds), while the fixed constraint show a decrease from 92% (zero refolds) to only 3% (four refolds).

Refolds	**Domain Success**		**NT Success (relaxed)**		**NT Success (fixed)**	
	False	**True (Rate)**	**False**	**True (Rate)**	**False**	**True (Rate)**
0	0	51 (100%)	1	50 (98%)	4	47 (92%)
1	0	128 (100%)	16	112 (88%)	20	108 (84%)
2	0	123 (100%)	24	99 (80%)	35	88 (72%)
3	0	84 (100%)	31	53 (63%)	56	28 (33%)
4	1	31 (97%)	22	10 (31%)	31	1 (3%)
Total	**1**	**417 (100%)**	**94**	**324 (76%)**	**146**	**272 (65%)**

Table 4.2: Comparison of domain and nucleotide success counts for relaxed and fixed constraints per number of refolds, including totals for overall success rates.

Figure 4.1 further illustrates these results by differentiating nucleotide success based on successful domain-level outcomes for the relaxed (A) and fixed (B) constraint. Figures A3 and B3 illustrate the distribution of attempts required to achieve a successful nucleotide design. The general data in both scenarios clearly follow an exponential trend, with the majority of successful sequences generated on the first attempt. Beyond the first few attempts, the probability of success decreases significantly. The median number of attempts for successful domain-level designs is 2 for relaxed and 1 for fixed constraints.

Discussion

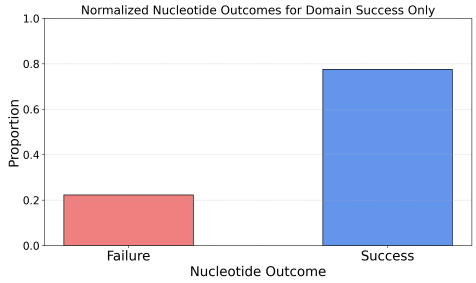
The analysis provided valuable insights into how the different components of the CocoPaths package work together in the design and translation of folding paths. The evaluation of folding paths of length six allowed for a comprehensive yet computationally feasible evaluation that effectively captured representative interactions across domain and nucleotide level designs. The high success rate of 99.8% at the domain level indicates that the designs generated by CocoPaths behave as intended under domain-level kinetics, demonstrating that the framework works not only from a purely logical point of view, but also when the sequence is simulated during cotranscriptional folding, where the kinetics of the sequence influence success.

A closer examination of the relaxed and fixed constraint variants reveals differences in their impact on nucleotide-level design feasibility. As shown in table 4.2, the relaxed constraint approach achieves an overall nucleotide-level success rate of 76%, significantly outperforming the fixed constraint, which achieves only 65%. This result indicates that the flexibility provided by the relaxed constraint greatly simplifies the design of successful nucleotide sequences. In contrast, the strict enforcement of pairing requirements in the fixed constraint scenario significantly limits the range of achievable designs. Although this

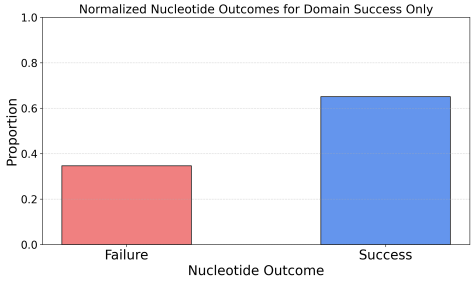
Relaxed constraint

Fixed constraint

A1



B1



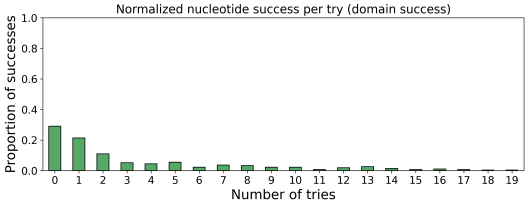
A2

		NT success		
		False	True	Total
Domain success	False	1	0	1
	True	93	324	417
	Total	94	324	418

B2

		NT success		
		False	True	Total
Domain success	False	1	0	1
	True	145	272	417
	Total	146	272	418

A3



B3

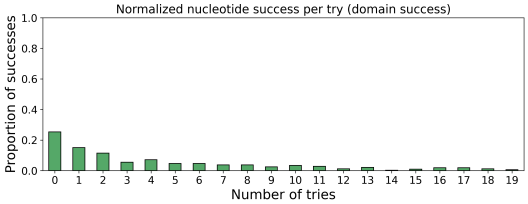


Figure 4.1: Success rates and design efficiency of coco-translatable paths under relaxed (A) versus fixed (B) folding path constraints. Panels A1 and B1 compare the fraction of paths achieving domain-level success with the proportion of nucleotide-level designs that reach over 50% occupancy of the target structure at specific transcript lengths. Panels A2 and B2 provide the corresponding counts from A1 and B1, respectively. Panels A3 and B3 show the distribution of successful nucleotide designs based on the number of attempts required to obtain a successful sequence, illustrating differences in design complexity between the two constraint conditions.

result is consistent with initial expectations, it underscores the importance of carefully selecting constraint types according to design objectives.
The data in table 4.2 suggests that the complexity of the folding path, as measured by the number of refolds, affects the success rates. Domain level success rates remain relatively stable across refold counts, indicating that complex domain level folding pathways can be successfully implemented. In contrast, overall nucleotide level success rates, while high at first, decrease with higher refold counts, suggesting that more complex folding pathways pose additional challenges at the nucleotide level. An important point to note is that as the number of refolds increases, the designs become more difficult to achieve. This is likely due to the added complexity introduced with each refold. However, the lower success rate may also result from constraints being easier to fulfill in paths with fewer refolds. In the case of the relaxed constraint unpaired regions are treated as wildcards during verification. These regions are ignored as long as they do not violate the specified pairing constraints. Designs with a low number of refolds often have easier pairing constraints to fulfill. This results in simpler target designs because more regions are ignored in the validation process. In contrary, the fixed constraint were unpaired regions must stay unpaired, the success rate drops significantly with an increased refold count. Interestingly, the fixed constraint still performed surprisingly well at lower refold counts, indicating that the strict enforcement of unpaired regions remains manageable at low refold numbers. While it was expected that at high refold numbers the difference between relaxed and fixed constraints would be minimal, the data indicate that the overall success rate for the fixed constraint remains significantly lower. At high refold numbers, most of regions in the aCFP are expected to be paired, with few unpaired regions, which should, in theory, lower differences between the constraints. However, the requirement that spacer domains remain unpaired under the fixed constraint may contribute to the reduced success rate observed at high refold numbers. The exponential trend observed in Figure 4.1A3/B3 illustrates the efficiency of the nucleotide design process. Under both constraints, most successful sequences are generated within the first few tries, demonstrating the ability of the heuristic to quickly converge to successful results. In particular, the median number of attempts required for successful domain-level designs is 2 and 1 for relaxed and fixed constraints, respectively. Section 3.3.2 previously highlighted that lower scoring sequences do not always correspond to better simulated folding paths. This suggests that nucleotide generation sometimes encounters suboptimal solutions or dead ends. Developing an improved objective function could potentially improve design success, although at the expense of increased computational complexity.

Overall, the analysis shows that the CocoPaths package and its integrated software tools provide a robust foundation for designing cotranscriptional folding paths and generating nucleotide sequences that satisfy them. The theoretical design framework proposed in chapter 2 has demonstrated its effectiveness in designing cotranscriptional folding paths and related sequences. By incorporating the domain level as an intermediate layer, the framework provides a robust platform for validating and optimizing designs early in the process. This intermediate step provides valuable insight into whether a design is achievable with current parameters, allowing adjustments to domain lengths or

transcription rates before proceeding to the nucleotide level.

5 Discussion and outlook

This thesis set out to address the challenges in designing RNA sequences that conform to specific cotranscriptional folding paths. Allowing the creation of RNA sequences that exhibit specific structural confirmations during transcription. Using domains as an abstraction layer bridging the gap between the specification layer, where only structural rearrangements in the form of an abstract cotranscriptional folding path are specified and the nucleotide level, where theoretical and experimental analysis can occur to verify the results. The main focus on this thesis lies on the design and verification of domain-level sequences in the context of cotranscriptional folding. While also providing a tool to translate those domain-level designs into the nucleotide level. All of this cumulated in the software package CocoPaths, which consists of three fundamental softwares for folding path design and verification: **(I) CocoPath** a tool to design domain-level sequences based on an abstract cotranscriptional folding path,**(II) CocoSim** a cotranscriptional folding simulator on the domain-level, allows for verification and optimization of the domain-level sequence, **(III) CocoDesign** which translates the information gathered on the domain-level consisting of the domain-level sequence and the domain-level folding path, to generate a nucleotide sequence which exhibits a cotranscriptional folding path as indicated by the domain-level folding path but on the nucleotide level.

5.1 CocoPath

For CocoPath, a design framework has been developed to generate domain-level sequences that exhibit a specific folding path. A set of universal rules for enumerating single-stranded domain-level folding was established, and based on these rules, a rate-independent model of cotranscriptional folding at the domain-level was created. An aCFP is coco-translatable using the sls-scheme introduced in this thesis if it fulfills three criteria: assignable, consistent and traversable. Through the controlled addition of short domains in the form of toeholds and clamps, CocoPath generates a domain-level sequence that satisfies the abstract cotranscriptional folding path on which it is based. Although CocoPath establishes a solid framework for domain-level sequence generation, certain topics remain to be addressed.

5.1.1 sls-translate paths and coco-translatability

While the sls-design scheme defines a theoretical framework for translating aCFPs into domain-level sequences, not all sls-translatable paths are currently realizable in practice. The implementation developed in this thesis, as discussed throughout and analyzed in

detail in chapter 4, defines a stricter subset of sls-translatable paths referred to as coco-translatable. Coco-translatability captures those aCFPs that are not only sls-translatable in principle, but also compatible with the current translation algorithm. This subset is characterized by three verifiable properties: assignability, consistency and traversability. While the sls-scheme allows for complex domain-level designs, the current implementation uses a fixed, non-retroactive edge-weighting strategy. This restriction simplifies the consistency check and translation procedure but limits the types of refolding hierarchies that can be represented. As a result, some folding paths that are theoretically sls-translatable cannot currently be realized using CocoPath because they would require retroactive changes to previously assigned edge weights. Something the current algorithm does not permit.
The advantage of this restriction, however, is that the coco-translatable subset is precisely defined and implementable at the domain level. The framework ensures that for any coco-translatable aCFP, the generated sequence will reflect the correct folding hierarchy and dynamics.
Expanding beyond coco-translatability, by allowing some form of retroactive weight adjustment, remains a promising direction for future work. Such developments could bring CocoPath closer to realizing the full potential of the sls-design scheme.

Pseudoknots

The primary limitation of this work, as previously discussed, lies in the exclusion of pseudoknots from both the design framework and the CocoPaths package. Pseudoknots are an important structural feature of many naturally occuring RNAs as well as an important feature of synthetic RNAs. In principle the design framework mentioned in chapter 2 which is implement in CocoPath, could be extended to handle and produce domain-level sequences, where pseudoknots are allowed. The inclusion of pseudoknots would primarily limit the verification process on the domain-level while also complicating the verification on the nucleotide level. CocoSim, the simulator for domain-level cotranscriptional folding paths, is built upon Peppercorn to explore the reaction space and calculate transition rates between secondary structures in the ensemble. Since Peppercorn works in a pseudoknot free space and lacks the ability to account for pseuodknots, CocoSim cannot verify domain-level designs that incorporate pseudoknots. Without the verification of the domain-level folding path further translation into a nucleotide sequence is not recommended. Integrating pseudoknots into the CocoPaths package would require a major overhaul of the Peppercorn package, building on its existing code to enable pseudoknots. This effort would also require the development of a novel algorithm capable of enumerating reactions involving pseudoknots, adding further complexity to the verification process.

Sequence length

Theoretically, the design framework outlined in chapter 2 can generate domain-level sequences based on an aCFP of arbitrary length. However, this is not practical. The nature of the design requires the addition of short domains to control folding and refolding

events, which causes the sequence to grow progressively longer with each additional folding or refolding event. As the sequence length increases, the RNA's internal refolding kinetics slow significantly. Longer helices make refolding events increasingly difficult, requiring longer timer domains to facilitate these events. This, in turn, further increases the sequence length. Aside from the kinetic challenge, longer sequences also increase the complexity of converting the domain sequence into a nucleotide sequence, as potential unintended interactions with the timer domains must be considered. A potential solution to mitigate the need for longer timer domains is the incorporation of pausing sites [49]. These sites allow RNA polymerase to pause temporarily, giving upstream regions sufficient time to fold into their minimum free energy structures before transcription resumes. The use of pausing sites offers a promising alternative to simply lengthening the sequence, as it provides additional time for the upstream sequence without the need for further lengthening. Further research is needed to explore the feasibility of pausing sites in this context. Experimental setups for RNA sequence analysis may require different pausing sequences and configurations, which must be carefully considered and tailored to specific applications.
Further research is needed to identify strategies for reducing the overall length of the resulting nucleotide sequence. As noted above, timer domains offer the greatest opportunity for length reduction. In contrast, other domain types, such as short and long domains, derive their functionality from their respective domain lengths, so reducing or increasing their intended domain lengths could result in the loss of their intended functions. This would undermine the function of the designed domain sequence. Another potential approach is to exclude certain clamps from the domain sequence. While toeholds are fundamental to any refolding event, clamps are primarily included to mitigate potential reverse reactions. Depending on the specific refolding event, some clamps could be removed from the sequence. However, this should be done with caution and validated with CocoSim before proceeding to nucleotide level design. Removing clamps has the potential to improve the overall kinetics of the RNA sequence by reducing it's overall length. Until a generalizable approach is developed, clamp removal should be performed on an individual basis for each domain sequence.

5.2 CocoSim

CocoSim is a tool for simulating cotranscriptional folding pathways, which is essential for validating the domain-level sequences resulting from CocoPath. The primary goal of CocoSim is to provide a platform for modeling and visualizing cotranscriptionally interesting designs at the domain-level, thus allowing the designs to be verified before progressing further. Verifying designs at the domain-level is an essential step in the design process to ensure a successful final nucleotide sequence. Built on the enumerator from Peppercorn [30], CocoSim employs a cyclic approach, similar to DrTransformer, to simulate the cotranscriptional folding path of an input sequence. The process involves five key steps: **(i)** expansion of the domain sequence, **(ii)** enumeration of the system, **(iii)** adjusting the occupancies in the system, **(iv)** simulation, and **(v)** pruning of unoccupied

structures.
During the simulation, CocoSim simulates the different conformations that the domain-level sequence can adopt using a chemical reaction network. Each conformation is represented as a vertex, while the kinetic rates of transitions between conformations are depicted as directed edges. By utilizing a modified version of the enumerate function from Peppercorn, which is restricted to unimolecular reactions, the reaction space is enumerated. This allows CocoSim to explore potential new secondary structures as domains are appended to the sequence, updating the structural ensemble within the chemical reaction network. Currently, the kinetic rates between secondary structures are based on DNA derived parameters. This stems from Peppercorn's original purpose of enumerating DNA strand displacement reactions. While this rate model is not entirely accurate for RNA, it does not significantly undermine the results. The dominant factor influencing kinetic rates is the length of the domains, making the simulations sufficiently reliable for modeling domain-level RNA sequences. As demonstrated in Section 4.2, predictions regarding whether a sequence is designable at the domain-level show a strong correlation with success at the nucleotide level. This suggests that despite the use of a DNA based rate model, CocoSim provides meaningful insight into the cotranscriptional folding paths of domain-level sequences.

5.3 CocoDesign

Once the domain-level sequence has been verified and confirmed to satisfy the aCFP from which it was derived, the final step in the design process is performed using CocoDesign. CocoDesign uses domain-level information, including the domain-level sequence and folding path, to generate a nucleotide sequence that mirrors the cotranscriptional folding path observed at the domain-level. The software is built on the Infrared package [47], which creates a sampling space for viable nucleotide sequences. This sampling space is defined by two constraints that impose domain-level properties on the nucleotide sequence. Using a Monte Carlo simulation approach, nucleotide sequences are sampled from this space and evaluated based on the developed objective function. The Monte Carlo method introduces randomness into the sampling process, reducing the likelihood of converging to a local optimum rather than the global one. If a newly sampled sequence improves the score of the current sequence, it is accepted. Otherwise, it is accepted with a probability that depends on the score difference, allowing exploration of potentially better solutions. While the current version of CocoDesign produces high quality nucleotide sequences, its effectiveness is limited by the objective function. As discussed in section 3.3.3, the success of a resulting nucleotide sequence does not consistently correlate with the score provided by the objective function. This inefficiency sometimes requires generating multiple designs and selecting the most appropriate one. Despite these limitations, the current iteration of the objective function strikes a balance between accurately reflecting the intended design and minimizing computational cost. This balance allows CocoDesign to function as a practical and efficient tool, offering a trade-off between precision and speed in the RNA design process.

5.4 Analysis

While the primary focus of this thesis is on theoretical and logical topics, an internal validation of the system was deemed necessary. The CocoPaths package operates across three distinct levels of abstraction for RNA, where each level fulfills a specific function. Although the software tools within the CocoPaths package cover at most two levels of abstraction, an analysis was performed that connects all three levels. The analysis follows the journey from the aCFP as input to the nucleotide sequence, with the goal of investigating how these levels interact and whether they produce usable nucleotide sequences. For this validation, a set of 418 coco-translatable aCFPs with a length of six was generated. These aCFPs were translated into domain sequences using CocoPath, and after validation with CocoSim, a nucleotide sequence was designed based on the domain-level information. The translation to the domain-level is deterministic, but the subsequent process of generating nucleotide sequences is heuristic. This means that the same input can result in nucleotide sequences of varying quality, with some performing better than others. Based on the observations discussed in section 3.3.3, where optimization occasionally results in unfavorable designs, the translation process was repeated 20 times. The goal of these replications was to identify successful nucleotide sequences.

The analysis demonstrated that the interplay between the layers is effective and that each layer fulfills its intended purpose. Out of the 418 designable paths, 417 were successfully translated and verified at the domain-level. This high success rate is consistent with expectations, as all designable paths should yield a valid domain-level sequence. These sequences satisfy the structural constraints at specific lengths when evaluated in the context of the minimum free energy structure. However, the key factor for success remains whether the sequence adopts the intended structures during the transcription process. The dynamic nature of transcription can sometimes prevent a sequence from folding into the intended structure, even when the structure represents the minimum free energy configuration. This mismatch may cause the sequence to adopt alternative conformations instead. Consequently, some aCFPs that are theoretically designable fail during domain-level verification. It is important to note that all domain-level designs and simulations were performed using default parameters. These parameters include fixed domain lengths and transcription speeds. The key takeaway from this analysis is that the domain level, as used in this work, serves as an effective intermediary between the abstract specification and the precise nucleotide level. It allows fine-tuning of parameters such as domain lengths and transcription speed to achieve robust results at the domain level before moving forward. Performance at this intermediate stage provides an early indication of the feasibility of the nucleotide-level design, allowing a quick evaluation of whether certain designs are worth pursuing before committing to the more time-consuming process of generating a complete nucleotide sequence.

5.5 Potential applications

The ability to create and verify domain-level designs with the CocoPaths package could lead to deeper insights into the kinetics of RNA folding. CocoPaths provides the basis for generating complex folding paths that can theoretically be observed by corresponding nucleotide sequences. Although these nucleotide sequences can be evaluated using established cotranscriptional folding simulators[24, 29], experimental validation of such nucleotide sequences with complex folding paths in vitro would be of great interest. Such validation could lead to improvements and new insights in kinetic models.
The use of domains to design complex nucleic acid structures is becoming increasingly common, particularly in nucleic acid nanotechnology [42, 39]. With the introduction of CocoSim, we provide a tool to verify that a designed domain-level sequence folds into the intended structure under cotranscriptional conditions. This allows an a priori analysis of domain-level sequences to identify potential kinetic traps and assess the dynamic accessibility of structural intermediates. By tuning design parameters such as domain length and transcription rate, CocoSim facilitates the optimization of folding behavior before proceeding to the nucleotide level.
To translate domain-level designs into nucleotide sequences, the CocoPaths package includes CocoDesign. This tool uses the information extracted from the domain-level specification, such as domain identities, complementarity constraints, and the validated folding path, to generate nucleotide sequences that are compatible with the intended cotranscriptional folding behavior.

5.6 Outlook

This thesis has introduced a novel approach for using the domain-level to design and validate sequences that follow specific cotranscriptional folding paths. Resulting in the CocoPaths package presented in this thesis. While CocoPaths lays a solid foundation for the design process some potential improvements can be made in the future. Given the increasing interest in understanding pseudoknots, the development of more advanced and efficient software for modeling systems with pseudoknots is anticipated. Expanding the CocoPaths package to include pseudoknots as legal secondary structures would significantly broaden its design capabilities. Allowing CocoPaths to handle pseudoknots is primarily limited by the software behind Peppercorn. While Peppercorn was originally intended to enumerate strand displacement reactions, incorporating pseudoknots would enhance its functionality and expand its use cases, as pseudoknots are a common and valuable design element for constructing complex nucleic acid structures. Thus, both CocoPaths and Peppercorn would benefit from the inclusion of pseudoknots. The growing interest in kinetically relevant RNA designs at the domain-level highlights another area for development: the need for domain-level kinetic parameters. Incorporating such parameters would allow Peppercorn to simulate and calculate kinetic rates between RNA structures more accurately. This would also enhance CocoSim, enabling more realistic predictions of folding dynamics and better validation of domain-level designs.

Another important limitation addressed in this thesis is the distinction between sls-translatability and coco-translatability. While the sls-design scheme defines a broad space of theoretically translatable folding paths, the current implementation only realizes a subset of these: the coco-translatable paths. This subset is defined by paths that satisfy the algorithm's current constraints, particularly the non-retroactive assignment of edge weights during the consistency check. As shown in chapter 2, this prevents the translation of certain theoretically valid aCFPs that require more flexible, hierarchical binding logic. In future work, more generalized and adaptive translation algorithms may overcome this limitation, allowing CocoPaths to capture a greater portion of the sls-translatable design space. Nonetheless, this thesis provides a solid foundation for domain-level design in cotranscriptional folding. The design framework introduced in Chapter 2 opens up new possibilities for creating sequences, while the ability to simulate cotranscriptional folding paths at the domain-level provides a means to verify and optimize these designs. Addressing the challenges identified in this thesis and incorporating advancements in software and kinetic modeling will further enhance RNA design capabilities. These developments will pave the way for designing increasingly complex and functional RNA structures, building on the contributions made in this work.

Bibliography

[1] J S Mattick. "Non-coding RNAs: the architects of eukaryotic complexity". en. In: *EMBO Rep.* 2.11 (Nov. 2001), pp. 986–991.

[2] N B Leontis and E Westhof. "Geometric nomenclature and classification of RNA base pairs". en. In: *RNA* 7.4 (Apr. 2001), pp. 499–512.

[3] Tianbing Xia et al. "Thermodynamic Parameters for an Expanded Nearest-Neighbor Model for Formation of RNA Duplexes with WatsonCrick Base Pairs". In: *Biochemistry* 37.42 (Oct. 1998), pp. 14719–14735. ISSN: 1520-4995. DOI: 10.1021/bi9809425. URL: http://dx.doi.org/10.1021/bi9809425.

[4] Ronny Lorenz et al. "ViennaRNA Package 2.0". In: *Algorithms for Molecular Biology* 6.1 (Nov. 2011), p. 26. ISSN: 1748-7188. DOI: 10.1186/1748-7188-6-26. URL: https://doi.org/10.1186/1748-7188-6-26.

[5] Jacek Nowakowski and Ignacio Tinoco. "RNA Structure and Stability". In: *Seminars in Virology* 8.3 (1997), pp. 153–165. ISSN: 1044-5773. DOI: https://doi.org/10.1006/smvy.1997.0118. URL: https://www.sciencedirect.com/science/article/pii/S1044577397901189.

[6] Ling X. Shen, Zhuoping Cai and Ignacio Tinoco. "RNA structure at high resolution". In: *The FASEB Journal* 9.11 (Aug. 1995), pp. 1023–1033. ISSN: 1530-6860. DOI: 10.1096/fasebj.9.11.7544309. URL: http://dx.doi.org/10.1096/fasebj.9.11.7544309.

[7] David H. Mathews et al. "Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure". In: *Proceedings of the National Academy of Sciences* 101.19 (May 2004), pp. 7287–7292. ISSN: 1091-6490. DOI: 10.1073/pnas.0401799101. URL: http://dx.doi.org/10.1073/pnas.0401799101.

[8] Mirela Andronescu et al. "Efficient parameter estimation for RNA secondary structure prediction". In: *Bioinformatics* 23.13 (July 2007), pp. i19–i28. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btm223. URL: http://dx.doi.org/10.1093/bioinformatics/btm223.

[9] Ruth Nussinov et al. "Algorithms for Loop Matchings". In: *SIAM Journal on Applied Mathematics* 35.1 (July 1978), pp. 68–82. ISSN: 1095-712X. DOI: 10.1137/0135006. URL: http://dx.doi.org/10.1137/0135006.

[10] M. Zuker. "Mfold web server for nucleic acid folding and hybridization prediction". In: *Nucleic Acids Research* 31.13 (July 2003), pp. 3406–3415. ISSN: 1362-4962. DOI: 10.1093/nar/gkg595. URL: http://dx.doi.org/10.1093/nar/gkg595.

[11] David H Mathews and Douglas H Turner. "Prediction of RNA secondary structure by free energy minimization". In: *Current Opinion in Structural Biology* 16.3 (2006). Nucleic acids/Sequences and topology, pp. 270–278. ISSN: 0959-440X. DOI: https://doi.org/10.1016/j.sbi.2006.05.010. URL: https://www.sciencedirect.com/science/article/pii/S0959440X06000819.

[12] D. Mathews. "Using an RNA secondary structure partition function to determine confidence in base pairs predicted by free energy minimization." In: *RNA* 10 8 (2004), pp. 1178–90. DOI: 10.1261/RNA.7650904.

[13] John S McCaskill. "The equilibrium partition function and base pair binding probabilities for RNA secondary structure". In: *Biopolymers: Original Research on Biomolecules* 29.6-7 (1990), pp. 1105–1119.

[14] Z. Lu, Jason W. Gloor and D. Mathews. "Improved RNA secondary structure prediction by maximizing expected pair accuracy." In: *RNA* 15 10 (2009), pp. 1805–13. DOI: 10.1261/rna.1643609.

[15] Jiří Šponer et al. "RNA Structural Dynamics As Captured by Molecular Simulations: A Comprehensive Overview". In: *Chemical Reviews* 118.8 (Jan. 2018), pp. 4177–4338. ISSN: 1520-6890. DOI: 10.1021/acs.chemrev.7b00427. URL: http://dx.doi.org/10.1021/acs.chemrev.7b00427.

[16] Tao Pan and Tobin Sosnick. "RNA FOLDING DURING TRANSCRIPTION". In: *Annual Review of Biophysics* 35.Volume 35, 2006 (2006), pp. 161–175. ISSN: 1936-1238. DOI: https://doi.org/10.1146/annurev.biophys.35.040405.102053. URL: https://www.annualreviews.org/content/journals/10.1146/annurev.biophys.35.040405.102053.

[17] Thomas J Santangelo and Irina Artsimovitch. "Termination and antitermination: RNA polymerase runs a stop sign". en. In: *Nat. Rev. Microbiol.* 9.5 (May 2011), pp. 319–329.

[18] Fred Russell Kramer and Donald R Mills. "Secondary structure formation during RNA synthesis". In: *Nucleic acids research* 9.19 (1981), pp. 5109–5124.

[19] Lisa Muniz, Estelle Nicolas and Didier Trouche. "RNA polymerase II speed: a key player in controlling and adapting transcriptome composition". In: *The EMBO Journal* 40.15 (July 2021). ISSN: 1460-2075. DOI: 10.15252/embj.2020105740. URL: http://dx.doi.org/10.15252/embj.2020105740.

[20] M Behfar Ardehali and John T Lis. "Tracking rates of transcription and splicing in vivo". In: *Nature Structural amp; Molecular Biology* 16.11 (Nov. 2009), pp. 1123–1124. ISSN: 1545-9985. DOI: 10.1038/nsmb1109-1123. URL: http://dx.doi.org/10.1038/nsmb1109-1123.

[21] Tao Pan and Tobin Sosnick. "RNA FOLDING DURING TRANSCRIPTION". In: *Annual Review of Biophysics* 35.Volume 35, 2006 (2006), pp. 161–175. ISSN: 1936-1238. DOI: https://doi.org/10.1146/annurev.biophys.35.040405.102053. URL: https://www.annualreviews.org/content/journals/10.1146/annurev.biophys.35.040405.102053.

[22] Martha S Rook, Daniel K Treiber and James R Williamson. "Fast folding mutants of the Tetrahymena group I ribozyme reveal a rugged folding energy landscape11Edited by D. Draper". In: *Journal of Molecular Biology* 281.4 (1998), pp. 609–620. ISSN: 0022-2836. DOI: https://doi.org/10.1006/jmbi.1998.1960. URL: https://www.sciencedirect.com/science/article/pii/S002228369891960X.

[23] Daniel K Treiber and James R Williamson. "Exposing the kinetic traps in RNA folding". In: *Current Opinion in Structural Biology* 9.3 (1999), pp. 339–345. ISSN: 0959-440X. DOI: https://doi.org/10.1016/S0959-440X(99)80045-1. URL: https://www.sciencedirect.com/science/article/pii/S0959440X99800451.

[24] CHRISTOPH FLAMM et al. "RNA folding at elementary step resolution". In: *RNA* 6.3 (Mar. 2000), pp. 325–338. ISSN: 1355-8382. DOI: 10.1017/s1355838200992161. URL: http://dx.doi.org/10.1017/s1355838200992161.

[25] Eric C. Dykeman. "An implementation of the Gillespie algorithm for RNA kinetics with logarithmic time update". In: *Nucleic Acids Research* 43.12 (May 2015), pp. 5708–5715. ISSN: 1362-4962. DOI: 10.1093/nar/gkv480. URL: http://dx.doi.org/10.1093/nar/gkv480.

[26] Alexander P. Gultyaev, F.H.D. van Batenburg and Cornelis W.A. Pleij. "The Computer Simulation of RNA Folding Pathways Using a Genetic Algorithm". In: *Journal of Molecular Biology* 250.1 (1995), pp. 37–51. ISSN: 0022-2836. DOI: https://doi.org/10.1006/jmbi.1995.0356. URL: https://www.sciencedirect.com/science/article/pii/S0022283685703567.

[27] Kyozi Kawasaki. "Diffusion Constants near the Critical Point for Time-Dependent Ising Models. II". In: *Physical Review* 148.1 (Aug. 1966), pp. 375–381. ISSN: 0031-899X. DOI: 10.1103/physrev.148.375. URL: http://dx.doi.org/10.1103/physrev.148.375.

[28] Michael T Wolfinger et al. "Efficient computation of RNA folding dynamics". In: *Journal of Physics A: Mathematical and General* 37.17 (Apr. 2004), pp. 4731–4741. ISSN: 1361-6447. DOI: 10.1088/0305-4470/37/17/005. URL: http://dx.doi.org/10.1088/0305-4470/37/17/005.

[29] Stefan Badelt, Ronny Lorenz and Ivo L Hofacker. "DrTransformer: heuristic cotranscriptional RNA folding using the nearest neighbor energy model". In: *Bioinformatics* 39.1 (Jan. 2023), btad034. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btad034. eprint: https://academic.oup.com/bioinformatics/article-pdf/39/1/btad034/49002717/btad034.pdf. URL: https://doi.org/10.1093/bioinformatics/btad034.

[30] Stefan Badelt et al. "A domain-level DNA strand displacement reaction enumerator allowing arbitrary non-pseudoknotted secondary structures". In: *Journal of The Royal Society Interface* 17.167 (June 2020), p. 20190866. ISSN: 1742-5662. DOI: 10.1098/rsif.2019.0866. URL: http://dx.doi.org/10.1098/rsif.2019.0866.

[31] I. L. Hofacker et al. “Fast folding and comparison of RNA secondary structures”. In: *Monatshefte fr Chemie Chemical Monthly* 125.2 (Feb. 1994), pp. 167–188. ISSN: 1434-4475. DOI: 10.1007/bf00818163. URL: http://dx.doi.org/10.1007/BF00818163.

[32] R. M. Dirks. “Paradigms for computational nucleic acid design”. In: *Nucleic Acids Research* 32.4 (Feb. 2004), pp. 1392–1403. ISSN: 1362-4962. DOI: 10.1093/nar/gkh291. URL: http://dx.doi.org/10.1093/nar/gkh291.

[33] Michael Zuker and Patrick Stiegler. “Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information”. In: *Nucleic Acids Research* 9.1 (1981), pp. 133–148. ISSN: 1362-4962. DOI: 10.1093/nar/9.1.133. URL: http://dx.doi.org/10.1093/nar/9.1.133.

[34] Michael Schnall-Levin, Leonid Chindelevitch and Bonnie Berger. “Inverting the Viterbi algorithm: an abstract framework for structure design”. In: *Proceedings of the 25th International Conference on Machine Learning*. ICML ’08. Helsinki, Finland: Association for Computing Machinery, 2008, pp. 904–911. ISBN: 9781605582054. DOI: 10.1145/1390156.1390270. URL: https://doi.org/10.1145/1390156.1390270.

[35] CHRISTOPH FLAMM et al. “Design of multistable RNA molecules”. In: *RNA* 7.2 (Feb. 2001), pp. 254–265. ISSN: 1355-8382. DOI: 10.1017/s1355838201000863. URL: http://dx.doi.org/10.1017/s1355838201000863.

[36] Michael Schmitz and Gerhard Steger. “Description of RNA Folding by "Simulated Annealing">”. In: *Journal of Molecular Biology* 255.1 (1996), pp. 254–266. ISSN: 0022-2836. DOI: https://doi.org/10.1006/jmbi.1996.0021. URL: https://www.sciencedirect.com/science/article/pii/S0022283696900212.

[37] Peter Steffen et al. “RNAshapes: an integrated RNA analysis package based on abstract shapes”. In: *Bioinformatics* 22.4 (Dec. 2005), pp. 500–503. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btk010. URL: http://dx.doi.org/10.1093/bioinformatics/btk010.

[38] A. Nishikawa, M. Yamamura and M. Hagiya. “DNA computation simulator based on abstract bases”. In: *Soft Computing* 5.1 (Feb. 2001), pp. 25–38. ISSN: 1432-7643. DOI: 10.1007/s005000000062. URL: http://dx.doi.org/10.1007/s005000000062.

[39] Matthew R. Lakin et al. “Visual DSD: a design and analysis tool for DNA strand displacement systems”. In: *Bioinformatics* 27.22 (Oct. 2011), pp. 3211–3213. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btr543. URL: http://dx.doi.org/10.1093/bioinformatics/btr543.

[40] Chenxiang Lin et al. “DNA tile based self-assembly: building complex nanoarchitectures”. In: *ChemPhysChem* 7.8 (2006), pp. 1641–1647.

[41] David Yu Zhang and Erik Winfree. “Control of DNA Strand Displacement Kinetics Using Toehold Exchange”. In: *Journal of the American Chemical Society* 131.47 (Dec. 2009), pp. 17303–17314. ISSN: 0002-7863. DOI: 10.1021/ja906987s. URL: https://doi.org/10.1021/ja906987s.

[42] Cody Geary et al. "RNA origami design tools enable cotranscriptional folding of kilobase-sized nanoscaffolds". In: *Nature Chemistry* 13.6 (May 2021), pp. 549–558. ISSN: 1755-4349. DOI: 10.1038/s41557-021-00679-1. URL: http://dx.doi.org/10.1038/s41557-021-00679-1.

[43] Andreas Mayer, Heather M Landry and L Stirling Churchman. "Pause & go: from the discovery of RNA polymerase pausing to its functional implications". In: *Current opinion in cell biology* 46 (2017), pp. 72–80.

[44] Eric Jones, Travis Oliphant, Pearu Peterson et al. "SciPy: Open source scientific tools for Python". In: (2001).

[45] Kyle E Watters et al. "Cotranscriptional folding of a riboswitch at nucleotide resolution". In: *Nature Structural amp; Molecular Biology* 23.12 (Oct. 2016), pp. 1124–1131. ISSN: 1545-9985. DOI: 10.1038/nsmb.3316. URL: http://dx.doi.org/10.1038/nsmb.3316.

[46] Courtney E Szyjka and Eric J Strobel. "Observation of coordinated RNA folding events by systematic cotranscriptional RNA structure probing". In: *Nature Communications* 14.1 (2023), p. 7839.

[47] Hua-Ting Yao et al. "Infrared: a declarative tree decomposition-powered framework for bioinformatics". In: *Algorithms for Molecular Biology* (2024). DOI: 10.1186/s13015-024-00258-2. URL: https://inria.hal.science/hal-04211173.

[48] Anda Ramona Tănasie et al. "DrForna: visualization of cotranscriptional folding". In: *Bioinformatics* 39.9 (Sept. 2023), btad555. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btad555. eprint: https://academic.oup.com/bioinformatics/article-pdf/39/9/btad555/51609745/btad555.pdf. URL: https://doi.org/10.1093/bioinformatics/btad555.

[49] Matthew H. Larson et al. "A pause sequence enriched at translation start sites drives transcription dynamics in vivo". In: *Science* 344 (2014), pp. 1042–1047. DOI: 10.1126/science.1251871.